A SAMPLING AND TRANSFORMATION APPROACH TO SOLVING RANDOM DIFFERENTIAL EQUATIONS

THESIS

Roger A. Erich, Second Lieutenant, USAF

AFIT/GAM/ENC/05-04

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government.

AFIT/GAM/ENC/05-04

A SAMPLING AND TRANSFORMATION APPROACH TO SOLVING RANDOM DIFFERENTIAL EQUATIONS

THESIS

Presented to the Faculty

Department of Mathematics and Statistics

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Applied Mathematics

Roger A. Erich, B.S.

Second Lieutenant, USAF

March 2005

AFIT/GAM/ENC/05-04

A SAMPLING AND TRANSFORMATION APPROACH TO SOLVING RANDOM DIFFERENTIAL EQUATIONS

Roger A. Erich, B.S.
Second Lieutenant, USAF

Approved:

/signed/	21 Mar 2005
Dr. Lawrence K. Chilton (Chairman)	date
/signed/	21 Mar 2005
Dr. Dennis W. Quinn (Member)	date
/signed/	21 Mar 2005
Maj. Anthony J. Pohl (Member)	date

AFIT/GAM/ENC/05-04

Abstract

This research explores an innovative sampling method used to conduct uncertainty analysis on a system with one random input. Given the distribution of the random input, X, we seek to find the distribution of the output random variable Y. When the functional form of the transformation $Y = g(X)$ is not explicitly known, complicated procedures, such as stochastic projection or Monte Carlo simulation, must be employed. The main focus of this research is determining the distribution of the random variable $Y = g(X)$ where $g(X)$ is the solution to an ordinary differential equation and X is a random parameter. Here, $y = g(X)$ is approximated by constructing a sample $\{X_i, Y_i\}$ where the X_i are not random, but chosen to be evenly spaced on the interval $[a, b]$ and $Y_i = g(X_i)$. Using this data, an efficient approximation $\hat{g}(X) \approx g(X)$ is constructed. Then the transformation method, in conjunction with $\hat{g}(X)$, is used to find the probability density function (pdf) of the random variable Y. This uniform sampling method and transformation method will be compared to the stochastic projection and Monte Carlo methods currently being used in uncertainty analysis. It will be demonstrated, through several examples, that the proposed uniform sampling method and transformation method can work faster and more efficiently than the methods mentioned.

Acknowledgements

The work contained in this thesis would not have been completed without the help and extensive knowledge of Dr. Chilton. His commitment to this research enabled me to learn vast amounts of material in the field of polynomial chaos and numerical analysis. In addition, I would like to thank Major Pohl and Dr. Quinn for volunteering their free time to read and improve my work.

I would also like to express deep gratitude for the love and unyielding support of my wife. She provided me with endless encouragement and was always willing to sacrafice her time to allow completion of this research. I thank my children for giving me all the hugs and kisses while I worked diligently on this paper. They also deserve recognition for giving up so much "daddy time" during the past eighteen months.

Finally, I would like to thank all of the faculty and fellow classmates for their knowledge and expertise they shared during my time at AFIT. I never thought that so much could be learned in such a short period of time.

I will never forget that, while sitting comfortably in my living room working on this thesis, men and women were putting their lives on the line defending our country's freedom and the freedom of others around the world. Without their sacrifice, none of this work would be possible. To those brave men and women, I salute you and thank you from the bottom of my heart.

Roger A. Erich

Table of Contents

Page

Abstract iv

Acknowledgements v

List of Figures viii

List of Tables xii

I. **Introduction** 1
- 1.1 **Background** 1
- 1.2 **Problem Definition** 5

II. **Methodology** 6
- 2.1 **The Functional Form of $g(X)$ is Known Explicitly** 6
  - 2.1.1 **Finding the PDF With the Transformation Method** 6
  - 2.1.2 **Finding the PDF With the Monte Carlo Method** 9
- 2.2 **The Functional Form of $g(X)$ is Not Known Explicitly** 10
  - 2.2.1 **Finding the PDF With The Transformation Method** 11
  - 2.2.2 **Finding the PDF With the Monte Carlo Method** 13

III. **Random Ordinary Differential Equations** 14
- 3.1 **Linear Example** 14
  - 3.1.1 **Approximating the Transformation $g : X \rightarrow Y$ Using the Stochastic Projection Method** 15
  - 3.1.2 **Approximating the Transformation $g : X \rightarrow Y$ Using the Uniform Sampling Method** . 23
  - 3.1.3 **Finding the Distribution With Approximate Transformation** 23
- 3.2 **Nonlinear Example** 26
  - 3.2.1 **Approximating the Transformation $g : X \rightarrow Y$ Using the Stochastic Projection Method** 27
  - 3.2.2 **Approximating the Transformation $g : X \rightarrow Y$ Using the Uniform Sampling Method** . 29

Page

IV. **Comparison of Hermite-chaos and Uniform Sampling Method Used to Approximate the Transformation** 31
4.1 **Accuracy and Computational Cost Evaluation of Approximated Transformation** 31
4.1.1 **Linear ODE** 31
4.1.2 **Nonlinear ODE** 35
4.2 **Accuracy Evaluation of Approximated PDF** . . . 39
4.2.1 **Linear ODE** 40
4.2.2 **Nonlinear ODE** 44

V. **Nonuniform Sampling Method** 50

VI. **Conclusion and Further Research** 53

Appendix A. **Mathematica Code for the Linear ODE to Approximate the Transformation** 55
A.1 **Hermite-chaos of Degree 4** 55
A.2 **Uniform Sampling** 58

Appendix B. **Mathematica Code for the Nonlinear ODE to Approximate the Transformation** 62
B.1 **Hermite-chaos of Degree 4** 62
B.2 **Uniform Sampling** 64
B.3 **Nonuniform Sampling** 68

Appendix C. **Mathematica Code for the CDF Transformation Method** 73

Appendix D. **Mathematica Code for Monte Carlo Simulation** . 84

Bibliography 89

List of Figures

Figure		Page
1.	Transformation $Y = \sin^2(X)$ Where X is Uniform(0,2π)	7
2.	PDF and CDF of X uniform(0,2π)	8
3.	PDF and CDF of $Y = \sin^2(X)$ with X uniform(0,2π)	9
4.	Histogram of $Y = \sin^2$(X) with X uniform(0,2π)	9
5.	$\hat{g}(X)$ Using Uniform Sampling and Interpolation With 100 Samples	11
6.	Approximate and Actual Transformation of $Y = g(X)$ Using Polynomial Chaos of Degree 8 With X Uniform(0,2π)	11
7.	PDF and CDF of $Y = \hat{g}(X)$ Using Transformation Method with X uniform(0,2π)	12
8.	Error Plot $Y = \lvert pdf - \hat{pdf} \rvert$ Using Transformation Method with X uniform(0,2π)	12
9.	PDF of $Y = \hat{g}(X)$ Using Monte Carlo With X uniform(0,2π) .	13
10.	Spring Mass Damper System	15
11.	Approximate (dashed) and Actual (solid) Transformation of $Y = g(\xi)$ Using Hermite-chaos of Degree 8.	21
12.	Error Plot $Y = \lvert g(\xi) - \hat{g}(\xi) \rvert$ Using Hermite-chaos of Degree 8. .	22
13.	Approximate (dashed) and Actual (solid) Transformation of $Y = tan(X)$ Using Polynomial Chaos of Degree 8 and Uniform Sampling Method With 80 Samples Where X Uniform(0,π)	22
14.	Actual and approximate Transformation $Y = g(X)$ Using Uniform Sampling Method With 500 Samples and Error Plot $Y = \lvert g(X) - \hat{g}(X) \rvert$	23
15.	PDF of Y Using "Input" Monte Carlo With 20,000 Runs . . .	24
16.	PDF of Y Using "Output" Monte Carlo With 20,000 Runs Where $\hat{g}(x)$ Obtained by Hermite-Chaos of Degree 8.	24
17.	CDF and PDF of Y Using Transformation Method With $\hat{g}(X)$ Obtained by Uniform Sampling With 512 Samples	25

Figure Page

18. Actual Transformation by Uniform Sampling Method Using 8192 Samples 27

19. Nonlinear ODE Transformation Using Hermite-chaos of Degree 2 and 4. 29

20. Nonlinear ODE Transformation Using Uniform Sampling Method With 256 Samples 30

21. PDF of Nonlinear ODE Using Uniform Sampling Method With 256 Samples and Monte Carlo Simulation With 50,000 Runs . . 30

22. Approximate (dashed) and Actual (solid) Transformation of $Y = g(X)$ Using Hermite-chaos of Degree 2 and 4 for Linear ODE. . 32

23. Log-Log Weighted Error Plot by Degree of Hermite Polynomial for Linear ODE. 33

24. Approximate (dashed) and Actual (solid) Transformation of $Y = g(X)$ Using Uniform Sampling Method With 32 and 64 Samples for Linear ODE. 33

25. Log-Log Weighted Error Plot by Number of Samples for Linear ODE. 34

26. Approximate (dashed) and Actual (solid) Transformation of $Y = g(X)$ Using Hermite-chaos of Degree 2 and 4 for Nonlinear ODE. 36

27. Log-Log Weighted Error Plot by Degree of Hermite Polynomial for Nonlinear ODE. 36

28. Approximate (dashed) and Actual (solid) Transformation of $Y = g(X)$ Using Uniform Sampling Method With 32 and 64 Samples for Nonlinear ODE. 37

29. Log-Log Weighted Error Plot by Number of Samples for Nonlinear ODE. 37

30. Approximate (solid) and Actual (dashed) $pdf(X)$ (Via Transformation Method) Using Hermite-chaos of Degree 2 and 4 for the Linear ODE. 40

31. Log-Log Error Plot for $pdf(X)$ (Via Transformation Method) by Degree of Hermite Polynomial for the Linear ODE. 41

Figure Page

32. Approximate (solid) and Actual (dashed) $pdf(X)$ (Via Transformation Method) Using Sampling Method With 32 and 64 Samples for the Linear ODE. 41

33. Log-Log Error Plot for $pdf(X)$ (Via Transformation Method) by Number of Samples for the Linear ODE. 42

34. Approximate (solid) and Actual (dashed) $pdf(X)$ (Via Monte Carlo Method) Using Hermite-chaos of Degree 2 and 4 for the Linear ODE. 43

35. Log-Log Error Plot for $pdf(X)$ (Via Monte Carlo Method) by Degree of Hermite Polynomial for the Linear ODE. 43

36. Approximate (solid) and Actual (dashed) $pdf(X)$ (Via Monte Carlo Method)Using Uniform Sampling With 32 and 64 Samples for the Linear ODE. 44

37. Log Log Error Plot for $pdf(X)$ (Via Monte Carlo Method) by Number of Samples for the Linear ODE. 44

38. Approximate (Solid) and Actual (dashed) $pdf(X)$ (Via Transformation Method) Using Hermite-chaos of Degree 2 and 4 for Nonlinear ODE. 45

39. Error Plot for $pdf(X)$ (Via Transformation Method) by Degree of Hermite Polynomial for Nonlinear ODE. 45

40. Approximate (solid) and Actual (dashed) $pdf(X)$ (Via Transformation Method) Using Uniform Sampling Method With 32 and 64 Samples for the Nonlinear ODE. 46

41. Error Plot for $pdf(X)$ (Via Transformation Method) by Number of Samples for Nonlinear ODE. 46

42. Approximate (solid) and Actual (dashed) $pdf(X)$ (Via Monte Carlo Simulation) Using Hermite-chaos of Degree 2 and 4 for the Nonlinear ODE. 47

43. Error Plot for $pdf(X)$ (Via Monte Carlo Simulation) by Degree of Hermite Polynomial for the Nonlinear ODE. 48

44. Approximate (solid) and Actual (dashed) $pdf(X)$ (Via Monte Carlo Simulation) Using Uniform Sampling Method With 32 and 64 Samples for the Nonlinear ODE. 48

Figure Page

45. Error Plot for $pdf(X)$ (Via Monte Carlo Simulation) by Number of Samples for the Nonlinear ODE. 49

46. CDF and Inverse CDF of N(0,1) Distribution. 51

47. Nonuniform Sample Generated Based on N(0,1) Distribution. . 51

48. Approximate (dashed) and Actual (solid) Transformation of $Y = g(X)$ Using Nonuniform Sampling Method With 35 and 67 Samples for Nonlinear ODE. 51

49. Log-Log Weighted Error Plot by Number of Nonuniform Samples for Nonlinear ODE. 52

List of Tables

Table		Page
1.	**Correspondence of Polynomials and Random Variables for Different Askey-Chaos ($N \geq 0$ is a finite integer)**	2
2.	**Summary of Methods Used to Find Transformations and Distributions**	6
3.	**Parameters for Linear Model ODE (1)**	19
4.	**Parameters for Nonlinear Model ODE (24)**	26
5.	**Comparison of Computational Time and Error Between Hermite-Chaos and Uniform Sampling Method Used to Approximate the Transformation for the Linear ODE**	34
6.	**Comparison of Computational Time and Error Between Hermite-Chaos and Uniform Sampling Method Used to Approximate the Transformation for the Nonlinear ODE**	38
7.	**Computational Time and Error of Hermite-Chaos Method With Precalculated Inner Products for the Nonlinear ODE**	38
8.	**Comparison of Error Between Hermite-chaos and Uniform Sampling Method Used to Approximate the PDF (Via Transformation Method) of the Linear ODE**	42
9.	**Comparison of Error Between Hermite-chaos and Uniform Sampling Method Used to Approximate the PDF (Via Transformation Method) of the Nonlinear ODE**	47
10.	**Comparison of Weighted Error Between Nonuniform and Uniform Sampling Methods Used to Approximate the Transformation of the Nonlinear ODE**	52

A SAMPLING AND TRANSFORMATION APPROACH TO SOLVING RANDOM DIFFERENTIAL EQUATIONS

I. Introduction

1.1 Background

Uncertainty analysis provides us with information regarding the uncertainty of system output when information, about inputs into the system, is known. If we know the distribution of the random inputs, X, into the system, then we seek to find the distribution of the output random variable Y. The complexity of this procedure depends on whether the functional form of the transformation $Y = g(X)$ is known explicitly or not. If it is explicitly known, standard statistical techniques can be applied. If it is not known, then other, more complicated procedures must be employed. The main focus of this research is determining the distribution of the random variable $Y = g(X)$ where $g(.)$ is the solution to an ordinary differential equation (ODE) and X is the set of random parameters. We define a random ODE as a differential equation where at least one parameter or initial condition is a random variable. Our goal is to introduce a more efficient method to quantify the uncertainty caused by random inputs into a system. This method, which we will refer to as the uniform sampling method, incorporates a very simple approach to seemingly complex problems. Here, $Y = g(X)$ is approximated by constructing a sample $\{X_i, Y_i\}$ where the X_i are not random, but chosen to be evenly spaced on a set determined by the range of X and $Y_i = g(X_i)$. Using this data, an efficient approximation $\hat{g}(X) \approx g(X)$ is constructed. Then an efficient alternative to Monte Carlo simulation, called the transformation method, is used in conjunction with $\hat{g}(X)$ to find the probability density function (pdf) of the random variable Y.

The concept of random input into a system, mainly engineering systems, has been studied extensively and many results have been attained. The stochastic projec-

tion method, described later in this text, provides a way to determine the distribution of the output random variable based on the random input. To build an expansion without knowing the form of the response, a random basis orthogonal to the distribution of the input uncertainty is typically selected. When polynomials are selected as the basis, the method is referred to as polynomial chaos [8]. The concept of Homogeneous Chaos was pioneered by Norbert Wiener, in 1938, involving the use of Hermite polynomials and the input random variable having a standard normal distribution [4]. Throughout the text, this will be referred to as Hermite-Chaos. In the generalized polynomial chaos, called the Askey-Chaos, the polynomials are not restricted to the Hermite polynomials and the random variables are not restricted to be Gaussian random variables [9]. When using the polynomial chaos method, it is standard practice to use polynomials that are orthogonal with respect to the pdf of the input random variable. In Table (1), the corresponding type of polynomial and their associated random variables are listed [9]. In this table, the support set is the region where the random variable is defined. We will employ Hermite-Chaos in this research since, by invoking the central limit theorem, the Gaussian distribution appears to be the most likely candidate for many physical applications [8].

	Random Variables	**Orthogonal Polynomials**	**Support**
Continuous	Gaussian	Hermite	$(-\infty, \infty)$
	Gamma	Laguerre	$[0, \infty)$
	Beta	Jacobi	[a,b]
	Uniform	Legendre	[a,b]
Discrete	Poisson	Charlier	$\{0, 1, 2, \dots\}$
	Binomial	Krawtchouk	$\{0, 1, 2, \dots N\}$
	Negative Binomial	Meixner	$\{0, 1, 2, \dots\}$
	Hypergeometric	Hahn	$\{0, 1, 2, \dots N\}$

Table 1: **Correspondence of Polynomials and Random Variables for Different Askey-Chaos ($N \geq 0$ is a finite integer)**

While Hermite-Chaos is useful in the analysis of stochastic processes, efforts have also been made to apply it to model uncertainty in physical applications [13]. These applications include structural fatigue, structural reliability, structural mechanics, linear structural dynamics, nonlinear random vibration, soil mechanics, soil-structure interaction, and simulation of non-Gaussian random fields [7]. For example, in [9], Hermite-Chaos is used to determine the cross-flow displacement of an elastically-mounted circular cylinder subject to stochastic inputs. The cylinder is free to move in the y-direction but not in the x-direction. This problem has two random inputs; damping of the cylinder and the stiffness. In this research, Hermite-Chaos will be used on linear and nonlinear random ordinary differential equations, with one random input, to find the functional form of the transformation $Y = g(X)$ when X is a standard normal random variable.

Another way to approximate the distribution of $Y = g(X)$ is through the use of Monte Carlo simulation. Here, a random sample from the input random variable is used to approximate the distribution of the output random variable. This can be slow with respect to computational time, but produces accurate results. In Xiu and Karniadakis [4], it was shown that Hermite-Chaos was substantially faster than Monte Carlo simulations for low dimensional stochastic inputs.

Modelling and simulation complexity challenges the state of computing and will continue to demand improvements in accuracy and efficiency [12]. Many of the major modelling and simulations efforts, supported by the U.S. Department of Energy (DOE), must address uncertainty and reliability in order to provide confidence levels for the results [12]. For example, reliability in predictions is crucial because ultimately these predictions can impact risk assessment decisions [12]. Predictive models of microbial cell and community functions are hindered by the lack of reliable and complete data for estimating kinetic parameters and identifying cell signaling pathway structure [12]. Answering this question is a key to the success of Genomes to Life as the results of combined experimental and computational studies on microbial sys-

tems are used to design strategies for bioremediation of the nuclear weapons complex, converting biomass to fuels, and carbon sequestration [12].

In a proposal submitted to the DOE from Pacific Northwest National Laboratory, a method referred to as Smart Sampling was introduced. Smart sampling implies the choice of an experimental design to specify a selection of input parameters that will best cover the possible outputs of the simulation [12]. A number of these sampling methodologies have been drawn from the design of physical experiments and modified for the design of computer experiments (This modification is needed because the notions of blocking, replication, and randomization do not apply to deterministic computer experiments) [12]. This smart sampling approach was used to determine efficient input into a computationally intensive computer model. The computer model can be thought of as a "black box" which can be evaluated but nothing else is known about the system. Then, by the sampling method, a functional approximation to the "black box" can be constructed and used to determine the distribution of the output. The overall impact of the project is the enabling of accurate approximation of mean behavior and uncertainty in complex computational models at a greatly reduced computational cost relative to Monte Carlo simulation [12].

In a recent research project involving a pitch and plunge airfoil with cubic and pentic structural parameters and uncertainties in the spring constant and initial pitch, a very efficient algorithm for determining the propagation of uncertainties in a highly nonlinear system was presented [6]. These uncertainties were allowed to propagate in a time dependent manner until a limit cycle oscillation or a dynamically stable solution was achieved [6]. The stochastic algorithm consisted of building an interpolating function in the stochastic domain through sampling and the use of a multivariate B-spline [6]. Advantages of this method were that expected values were never computed or stored and two orders of magnitude efficiency in computational time over a traditional Monte Carlo simulation were obtained [6]. Based on the PDFs predicted by the stochastic algorithm, a very rapid and accurate estimate of the probability of failure was obtained [6].

1.2 Problem Definition

While Hermite-Chaos and Monte Carlo simulation are effective methods of determining the distribution of the output random variable of some systems, there are problems when neither one produces satisfactory results. For complex models that consume considerable computational resources, the Monte Carlo method may be very costly and inefficient. Also, this approach does not readily provide information about the sensitivity of model outputs to specific parametric uncertainties [10]. In addition, polynomial chaos can fail when the transformation $g : X \rightarrow Y$ is not smooth. This occurs because polynomial chaos approximates the transformation using polynomial basis functions with global support. Classic results from polynomial approximation theory indicate that in order to approximate discontinuities well, piecewise polynomials must be used [11].

In the following discussion, the three methods, Hermite-Chaos, Monte Carlo simulation, and the uniform sampling method will be demonstrated and compared through the solution of both linear and nonlinear random ODEs. Computation time and comparison to the actual transformation will be analyzed using these three methods. Therefore, the validity of the uniform sampling method will be determined based on the results of these tests.

II. Methodology

There are several possible approaches for finding the distribution of a random variable Y, which depends on a random variable X when the distribution of X is known. These approaches can be divided into two cases. The first case is when the functional form of the transformation $g : X \to Y$ is known explicitly. The second case is when the functional form of $g(.)$ is not known explicitly, but $g(.)$ can be evaluated or at least approximated. This includes the situation when g is the solution to a differential equation but an explicit expression for g is unknown. In this case, a numerical method could be used to approximate g. These ideas will be explained in the next several sections and are summarized in Table (2).

	Approximate g(X)	**Find Distribution of Y**
Functional Form of g(X) is Known Explicitly	N/A	Transformation Method or Monte Carlo Simulation
Functional Form of g(X) is Not Known Explicitly	Stochastic Projection or Uniform Sampling and Interpolation	Transformation Method or Monte Carlo Simulation

Table 2: **Summary of Methods Used to Find Transformations and Distributions**

2.1 The Functional Form of $g(X)$ is Known Explicitly

In this section, we will use the transformation method and Monte Carlo Simulation to determine the pdf of the system output random variable Y. For the following examples $Y = sin^2(X)$ will be used as the known transformation $g(X)$.

2.1.1 ***Finding the PDF With the Transformation Method.*** Let X be a random variable with known probability density function (pdf) $f_X(x)$ with support $\mathcal{X}$ and cumulative distribution function (cdf) $F_X(x) = P(X \leq x)$. Also, let $Y = g(X)$ be a random variable that depends on X. If g and the pdf of X are known,

then the cdf of Y, $F_Y(y)$, can be determined as follows.

$$\begin{aligned} F_Y(y) = P(Y \leq y) &= P(g(X) \leq y) \\ &= P(\{x \in \mathcal{X} : g(x) \leq y\}) \\ &= \int_{\{x \in \mathcal{X} : g(x) \leq y\}} f_X(x)\, dx \end{aligned}$$

The pdf of Y is $f_Y(y) = \frac{d}{dy} F_Y(y)$. This approach for determining the distribution of Y will be referred to as the transformation method. As discussed in [3] this method requires g to be monotone over a partition of $\mathcal{X}$.

The following example from Casella and Berger [3] illustrates this method. Suppose X has a uniform distribution on $(0, 2\pi)$,

$$f_X(x) = \begin{cases} \frac{1}{2\pi} & 0 < x < 2\pi \\ 0 & \text{otherwise} \end{cases}$$

and let $g(x) = \sin^2(x)$ or

$$Y = \begin{cases} \sin^2(X) & 0 < x < 2\pi \\ 0 & \text{otherwise} \end{cases}$$

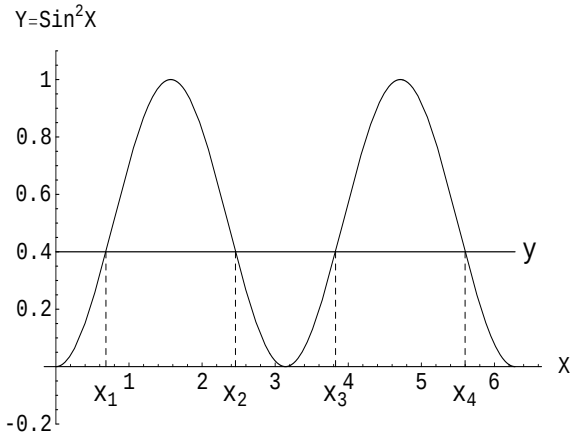


Figure 1: Transformation $Y = \sin^2(X)$ Where X is Uniform(0,2π)

Then, referring to Figure (1)

$$F_Y(y) = P(Y \leq y) = P(X \leq x_1) + P(x_2 \leq X \leq x_3) + P(x_4 \leq X)$$
$$= \cdots = 2\left[P(X \leq x_1) + P(x_2 \leq X \leq \pi)\right]$$

where $x_1 < x_2$ and x_1, x_2 are the solutions to $\sin^2(x) = y$, $0 < x < \pi$. Then x_1 is the smallest $x > 0$ such that $x = \arcsin(\sqrt{y})$ and $x_2 = \pi - x_1$. So

$$F_Y(y) = 2\left[\frac{x_1}{2\pi} + \frac{\pi - x_2}{2\pi}\right] = \cdots = \frac{2x_1}{\pi} = \frac{2}{\pi}\arcsin(\sqrt{y})$$

and

$$f_Y(y) = \frac{d}{dy}\left[\frac{2}{\pi}\arcsin(\sqrt{y})\right] = \frac{1}{\pi\sqrt{y(1-y)}}\,.$$

In Figure (2) the pdf and cdf of X are shown. In Figure (3) the pdf and cdf of $Y = \sin^2(X)$ are shown.

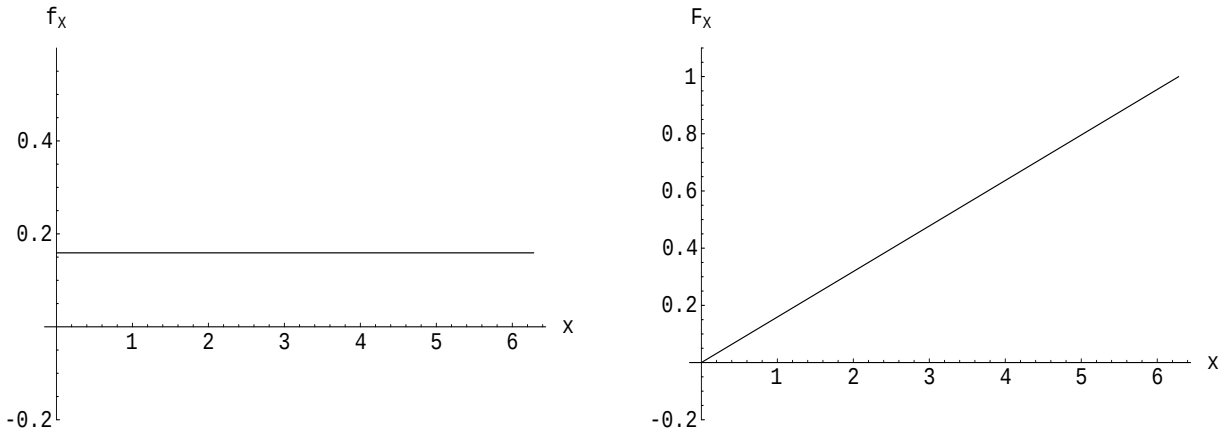


Figure 2: PDF and CDF of X uniform(0,2π)

This method relies on being able to partition X into regions on which $g(X)$ is monotone. It also requires that for a given Y, $X = g^{-1}(Y)$ can be computed. If either of these requirements are not met or is especially expensive to compute, this method will fail.

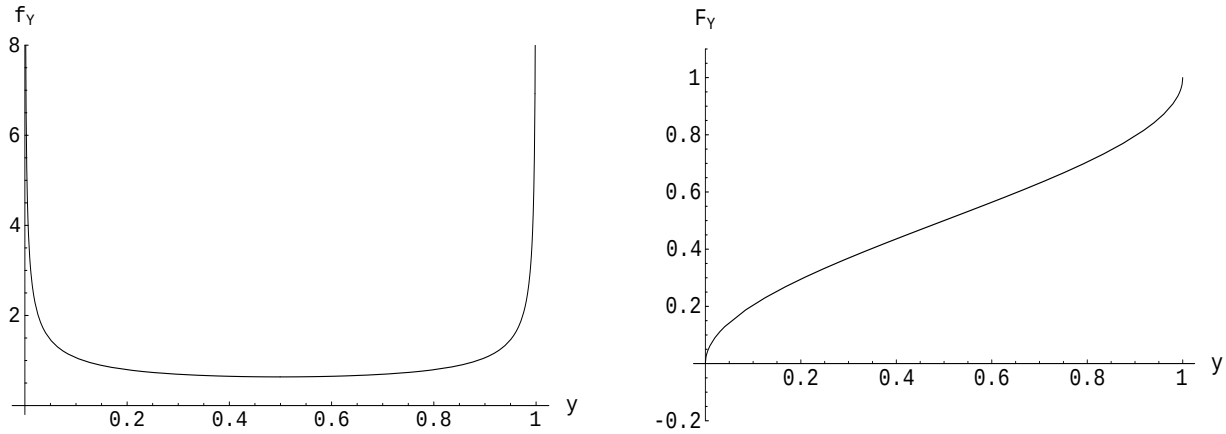


Figure 3: PDF and CDF of $Y = \sin^2(X)$ with X uniform(0,2π)

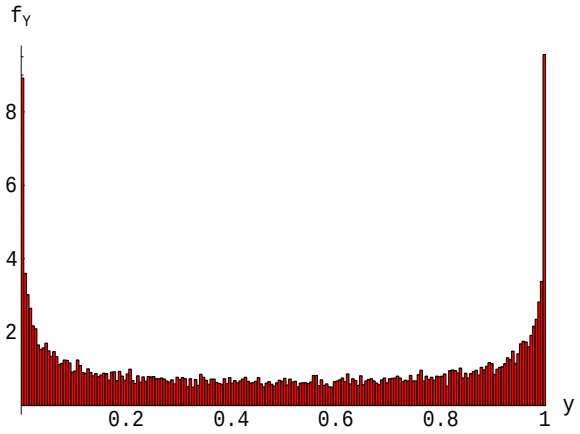


Figure 4: Histogram of $Y = \sin^2(\mathrm{X})$ with X uniform(0,2π)

2.1.2 ***Finding the PDF With the Monte Carlo Method.*** The transformation $y = g(x)$ can be evaluated for any x. Also assume the distribution of X is known or at least a method of generating a random sample from X is available. One way to approximate the pdf of $Y = g(X)$ is to generate a random sample $\{x_1, \ldots, x_n\}$ from X and then substitute the sampled values into $y = g(x)$, producing a random sample $\{y_1, \ldots, y_n\}$ from Y. A histogram of these sample values approximates the pdf of Y. The result of doing this with $g(x) = \sin^2 x$ and X uniform$(0, 2\pi)$ is shown in Figure (4), which matches the pdf in Figure (3) quite well. There are several other ways to approximate the pdf of Y given a random sample $\{y_1, \ldots, y_n\}$ which are out-

lined in [2]. Of course, these approaches can be applied to any sample $\{y_1, \ldots, y_n\}$ whether it was generated as $y = g(x)$ or not. These methods will be referred to as Monte Carlo methods because they are based on a random sample. While Monte Carlo simulation is the standard method in practice today, it has some limitations. It may be costly and inefficient especially when the system, under analysis, is computationally expensive. To obtain a useful pdf, the histogram must be smoothed. The transformation method, however, produces a more accurate and usable pdf without any smoothing or extra steps.

2.2 The Functional Form of $g(X)$ is Not Known Explicitly

In this section, we will use the transformation method and Monte Carlo Simulation to determine the pdf of the system output random variable Y. For the following examples $Y = sin^2(X)$ will be used as the unknown transformation $g(X)$. That is, the uniform sampling method and Hermite-chaos will be used to approximate the transformation $Y = sin^2(X)$.

In Table (2), there are two methods to obtain the approximation to the transformation $g(X)$, stochastic projection and uniform sampling. To demonstrate the uniform sampling method, we suppose $g(X) = sin^2(X)$ where X has a uniform distribution on $(0, 2\pi)$. This transformation can be approximated by uniformly sampling X over the interval $(0, 2\pi)$ producing a sample $\{x_1, \ldots, x_n\}$. Then let $y_i = g(x_i)$, producing a collection of points $\{x_i, y_i\}$ where $i = 1, \ldots, n$. Then we can interpolate g at $\{x_i, y_i\}$ producing the approximation $\hat{g}(X) \approx g(X)$. Figure (5) shows the piecewise linear interpolation of $sin^2(X)$ using 100 evenly spaced samples.

Next, we demonstrate the stochastic projection method for the transformation $g(X) = sin^2(X)$ where X has a Uniform distribution on $(0, 2\pi)$. From Table (1), we see that the Legendre polynomial is the correct orthogonal polynomial to use with random variables from the Uniform distribution. The result in Figure (6) shows that the stochastic projection method works quite well. The actual transformation, $g(X) = \sin^2(X)$, is shown as the solid plot while the approximated transformation,

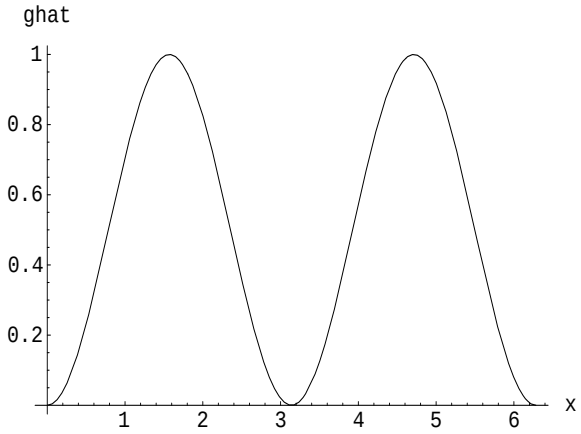


Figure 5: $\hat{g}(X)$ Using Uniform Sampling and Interpolation With 100 Samples

$\hat{g}(X)$, is the dashed plot. The stochastic projection method will be discussed in the next chapter.

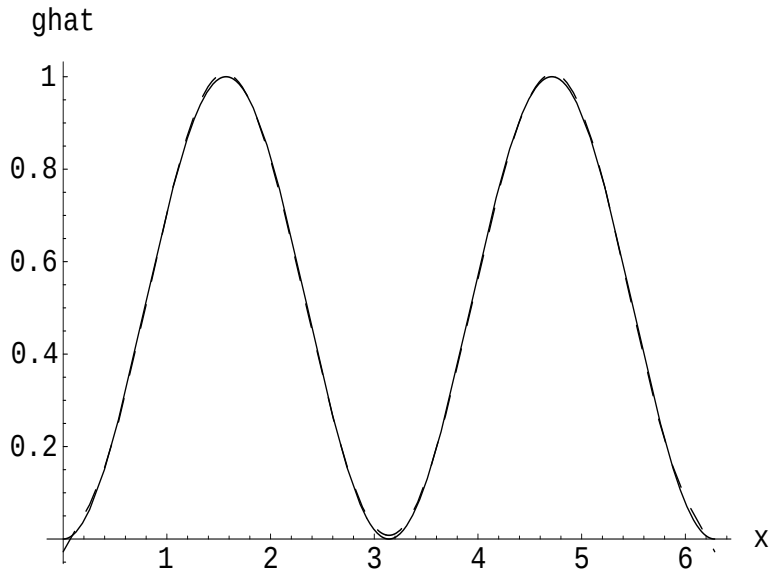


Figure 6: Approximate and Actual Transformation of $Y = g(X)$ Using Polynomial Chaos of Degree 8 With X Uniform(0,2π)

2.2.1 ***Finding the PDF With The Transformation Method.*** Once $\hat{g}(X)$ is obtained, the transformation method can be applied to get the cdf and pdf of the output random variable. The results for $\hat{g}(X)$ produced by sampling are found in

Figure (7). Notice that the approximation to $g(X)$ using sampling produces almost the exact pdf and cdf of the output random variable. The error plot shown in Figure (8), calculated as $|g(X) - \hat{g}(X)|$, verifies that the $\hat{g}(X)$ produced via sampling is a good approximation to the actual transformation. Similar results can be achieved using $\hat{g}(X)$ produced by stochastic projection.

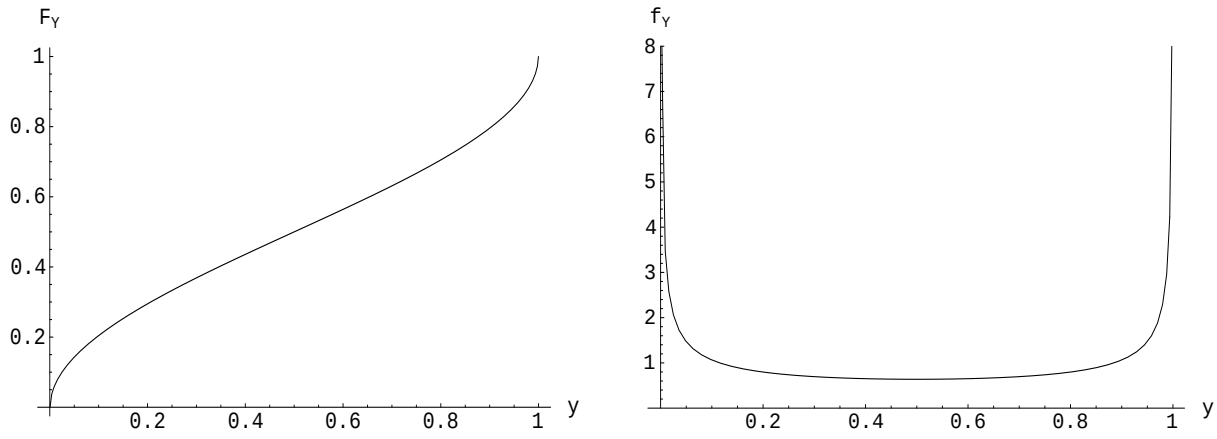


Figure 7: PDF and CDF of $Y = \hat{g}(X)$ Using Transformation Method with X uniform(0,2π)

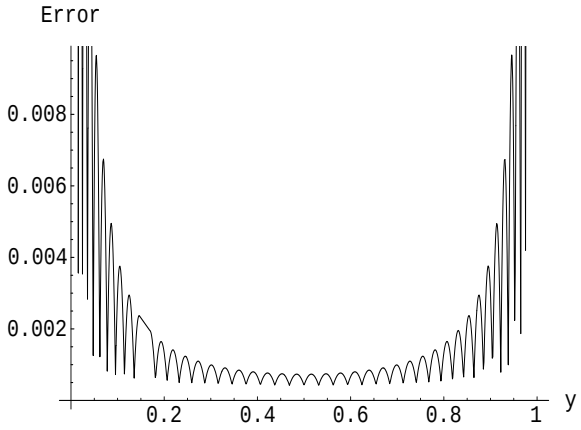


Figure 8: Error Plot $Y = |pdf - \hat{pdf}|$ Using Transformation Method with X uniform(0,2π)

2.2.2 ***Finding the PDF With the Monte Carlo Method.*** Using the Monte Carlo method described in section 2.1.2, we can approximate the pdf of the output random variable using $\hat{g}(X)$, found in section 2.2, with X having a uniform distribution on $(0, 2\pi)$. Once the sample is generated, a histogram is constructed to form the pdf. The pdf is shown in Figure (9) where $\hat{g}(X)$ was generated by the uniform sampling method with 100 samples. As expected, this function mirrors the exact pdf in section (2.1). Also, if evaluation of the function $g(x)$ is expensive, generating a sample $\{y_1, \ldots, y_n\}$ from $\{x_1, \ldots, x_n\}$ using Monte Carlo will also be expensive. Since $\hat{g}$ is typically a piecewise polynomial and can be evaluated efficiently, one approach would be to generate a random sample $\{x_1, \ldots, x_n\}$ using Monte Carlo and set $\hat{y}_i = \hat{g}(x_i)$ to efficiently create the sample $\{\hat{y}_i, \ldots, \hat{y}_n\}$. Then a histogram of this sample approximates the pdf of Y. Also, since the pdf of X is known, the transformation method using $\hat{g}$ could be used to better approximate the cdf and pdf of Y.

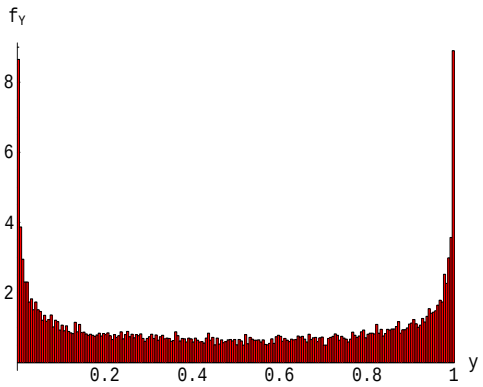


Figure 9: PDF of $Y = \hat{g}(X)$ Using Monte Carlo With X uniform(0,2π)

III. Random Ordinary Differential Equations

The polynomial chaos and uniform sampling methods can be applied directly to random ordinary differential equations (ODEs). In this case, the explicit functional form of the transformation is not known, but certain characteristics of the system, such as a rate of change, are known. Generally, systems under analysis exhibit random inputs. A random ODE is one where the differential operator $L[u(t); a]$ depends on a parameter set a and one or more elements of a are random variables. In addition, the initial or boundary conditions could be random variables. The methods used to approximate the transformation will be illustrated with two examples, a linear ODE and a nonlinear ODE.

3.1 Linear Example

The first example is the linear ODE

$$\begin{aligned} &m\ddot{u} + c\dot{u} + ku = 0 \\ &u(0) = 0 \\ &u'(0) = v \end{aligned} \tag{1}$$

which models the spring mass damper system found in Figure (10). Here k is the spring constant, m is the mass, and c is the damping coefficient. The solution to (1) is $u(t; m, c, \ldots)$, emphasizing that u depends on the independent variable and five parameters. This can be viewed as a transformation $Y(t) = g(X)$. If any component of X is random, then, for fixed t, $Y(t)$ is a random variable and the distribution of $Y(t)$ can be determined using the methods of Chapter 2 provided the distribution of the random components of X are known. Here, for $u(t; X)$, we have $X = (m, c, k, u(0), u'(0))$.

Of the methods presented in Chapter 2 for determining the distribution of $Y(t)$, we will focus on the case where the functional form of $g(.)$ is not known explicitly.

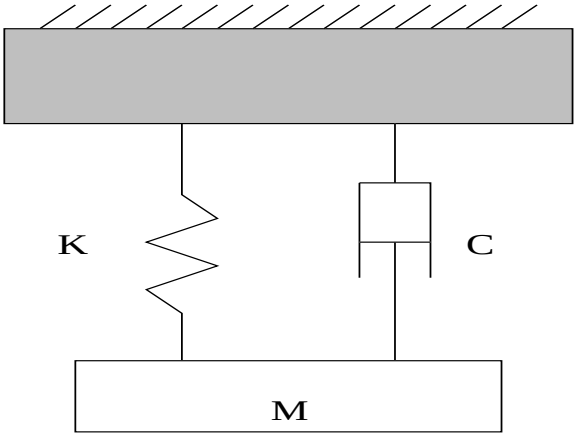


Figure 10: Spring Mass Damper System

Even though the model problem (1) can be solved explicitly, explicit solutions are not available for many problems of interest in applications.

3.1.1 ***Approximating the Transformation*** $g : X \to Y$ ***Using the Stochastic Projection Method.*** We begin with stochastic projection which is described extensively in Xiu and Karniadakis [4] and in Ghanem and Spanos [8]. The approximate transformation is represented by

$$u(t, \xi) = \sum_{i=0}^{P} x_i(t)\phi_i(\xi). \tag{2}$$

Given ϕ_i, we will show how to find x_i. How the functions ϕ_i are chosen will be discussed later in this chapter. The first step is to find $x_i(t)$ for $i = 0, \ldots, P$ where P is the order of the polynomial expansion. Given the model problem in Equation (1), we consider the case where there is one random parameter, which is the spring constant k. We let k be defined by

$$k = k(\xi) = \sum_{i=0}^{P} k_i\phi_i(\xi). \tag{3}$$

Next, we substitute Equation (3) and (2) into (1). This results in

$$\sum_{i=0}^{P} \ddot{x}_i(t)\phi_i(\xi) + \frac{c}{m}\sum_{i=0}^{P} \dot{x}_i(t)\phi_i(\xi) + \frac{1}{m}\sum_{i=0}^{P} k_i\phi_i(\xi)\sum_{j=0}^{P} x_j(t)\phi_j(\xi) = 0. \tag{4}$$

Then we multiply equation (4) by ϕ_l where $l = 0, \ldots, P$ and a weight function $w(x)$ and integrate on $[a, b]$. Letting $< f, g > \; = \int_a^b fgw\, dx$, we get

$$\begin{aligned} &\textstyle\sum_{i=0}^{P} \ddot{x}_i(t) < \phi_i, \phi_l > \; + \frac{c}{m}\sum_{i=0}^{P} \dot{x}_i(t) < \phi_i, \phi_l > \; + \frac{1}{m}\sum_{i=0}^{P}\sum_{j=0}^{P} k_i x_j(t) < \phi_i\phi_j, \phi_l > = 0 \\ &\text{for } l = 0, \ldots, P. \end{aligned} \tag{5}$$

If we choose ϕ so they are orthogonal with respect to $< ., . >$, then the result is

$$\begin{aligned} &[\ddot{x}_l(t) + \tfrac{c}{m}\dot{x}_l(t)] < \phi_l, \phi_l > \; + \tfrac{1}{m}\textstyle\sum_{i=0}^{P}\sum_{j=0}^{P} k_i x_j(t) < \phi_i\phi_j, \phi_l > = 0 \\ &\text{for } l = 0, \ldots, P. \end{aligned} \tag{6}$$

Dividing through by $< \phi_l, \phi_l >$, we get

$$\begin{aligned} &\ddot{x}_l(t) + \tfrac{c}{m}\dot{x}_l(t) \; + \tfrac{1}{m < \phi_l, \phi_l >}\textstyle\sum_{i=0}^{P}\sum_{j=0}^{P} k_i x_j(t) < \phi_i\phi_j, \phi_l > = 0 \\ &\text{for } l = 0, \ldots, P. \end{aligned} \tag{7}$$

To convert this system of second order ODEs to a system of first order ODEs, we let

$$\begin{aligned} &u_l = x_l(t) \\ &\dot{u}_l = \dot{x}_l(t) = v_l \\ &\dot{v}_l = \ddot{x}_l(t) = -\tfrac{c}{m}v_l - \tfrac{1}{m < \phi_l, \phi_l >}\textstyle\sum_{i=0}^{P}\sum_{j=0}^{P} k_i u_j < \phi_i\phi_j, \phi_l > \\ &\text{for } l = 0, \ldots, P. \end{aligned} \tag{8}$$

This gives us

$$z = \begin{bmatrix} u_0 \\ u_1 \\ . \\ . \\ u_P \\ v_0 \\ v_1 \\ . \\ . \\ v_P \end{bmatrix}, \ \dot{z} = \begin{bmatrix} v_0 \\ v_1 \\ . \\ . \\ v_P \\ -\frac{c}{m}v_0 - \frac{1}{m<\phi_0,\phi_0>}\sum_{i=0}^{P}\sum_{j=0}^{P} k_i u_j < \phi_i\phi_j, \phi_0 > \\ . \\ . \\ . \\ -\frac{c}{m}v_P - \frac{1}{m<\phi_P,\phi_P>}\sum_{i=0}^{P}\sum_{j=0}^{P} k_i u_j < \phi_i\phi_j, \phi_P > \end{bmatrix} \tag{9}$$

The resulting system that needs to be solved is

$$\dot{z} = Sz \tag{10}$$

where

$$S = \begin{bmatrix} & & 1 & . & 0 & 0 \\ & & 0 & . & 0 & 0 \\ & \mathbf{0} & . & . & . & . \\ & & . & . & 1 & 0 \\ & & 0 & . & 0 & 1 \\ -\frac{1}{m<\phi_0,\phi_0>}[\sum_{i=0}^{P} k_i < \phi_i\phi_0, \phi_0 >, ..., \sum_{i=0}^{P} k_i < \phi_i\phi_P, \phi_0 >] & & -\frac{c}{m} & . & 0 & 0 \\ & & 0 & -\frac{c}{m} & . & 0 \\ & & 0 & . & . & 0 \\ & & 0 & . & . & 0 \\ -\frac{1}{m<\phi_P,\phi_P>}[\sum_{i=0}^{P} k_i < \phi_i\phi_0, \phi_P >, ..., \sum_{i=0}^{P} k_i < \phi_i\phi_P, \phi_P >] & & 0 & . & 0 & -\frac{c}{m} \end{bmatrix} \tag{11}$$

To simplify the system, we choose the ϕ_j such that the inner product, $< \phi_j, \phi_j > = 1$ for all $j = 0, \ldots, P$. We also set each $k_i = 0$ for $i \geq 2$. Then the matrix S will be

$$S = \begin{pmatrix} 0 & I \\ -(\frac{k_0}{m}I + \frac{k_1}{m}A^{-1}B) & -\frac{c}{m}I \end{pmatrix} \tag{12}$$

Thus, the following description can be used to solve the system of ODEs to obtain the approximate transformation using stochastic projection. We define the following:

- a,b - the interval on which the random variable ξ is defined
- $\phi_i(\xi)$ - stochastic basis function
- m,c - parameters in ODE
- k - spring constant
- k_0 - mean value of k (will be denoted $\bar{k}$)
- k_1 - standard deviation of k (will be denoted $\tilde{k}$)
- v - initial velocity
- tf - final time
- $w(\xi)$ - the pdf of the distribution of the input random variable

We must construct the matrices A and B in the ODE system

$$\ddot{x} + c\dot{x} + (\frac{\bar{k}}{m}I + \frac{\tilde{k}}{m}A^{-1}B)x = 0 \tag{13}$$

where the matrix $A = [< \phi_i(\xi), \phi_j(\xi) >]$. These inner products are calculated by

$$< \phi_i(\xi), \phi_j(\xi) > = \int_a^b \phi_i(\xi)\phi_j(\xi)w(\xi)d\xi. \tag{14}$$

The matrix B is given by $B =< \xi\phi_i(\xi), \phi_j(\xi) >$ which is similar to the A matrix. These inner products are calculated by

$$< \xi\phi_i(\xi), \phi_j(\xi) > = \int_a^b \xi\phi_i(\xi)\phi_j(\xi)w(\xi)d\xi. \tag{15}$$

Once we have A and B, We use them to solve the system

$$\dot{z} = Sz \tag{16}$$

where S is the matrix defined by

$$S = \begin{pmatrix} 0 & I \\ -(\frac{\bar{k}}{m}I + \frac{\tilde{k}}{m}A^{-1}B) & -\frac{c}{m}I \end{pmatrix} \tag{17}$$

The result of solving Equation (16) gives us $x_k(t)$. The transformation can now be approximated using the equation $u(t, \xi) = \sum_{k=0}^{P} x_k(t)\phi_k(\xi)$.

We now look at the following example of the linear ODE described in (10). For this example, the polynomial chaos method described uses the one-dimensional Hermite polynomials in terms of standard, normally distributed random variables (ξ). We begin by defining $\phi_i(\xi)$ as the Hermite polynomial of degree i and $w(\xi)$ to be the pdf of the standard normal distribution. We also use the following parameters for the model problem (1)

a	b	m	$\bar{k}$	$\tilde{k}$	c	v	t
-6	6	10	10	2	4	25	10

Table 3: **Parameters for Linear Model ODE (1)**

For the function $\phi_i(\xi)$, we used the scaled Hermite polynomial

$$\phi_i(\xi) = \frac{H_i[\frac{\xi}{\sqrt{2}}]}{\sqrt{2^i i!}} \tag{18}$$

The scaling is needed so that $< \phi_i, \phi_i >= 1$. For the function $w(\xi)$, we used

$$w(\xi) = \frac{1}{\sqrt{2\pi}} e^{-\frac{\xi^2}{2}} \tag{19}$$

When $P = 8$, the matrices A and B are as follows

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{20}$$

$$B = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1.41421 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.41421 & 0 & 1.73205 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.73205 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 2.23607 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2.23607 & 0 & 2.44949 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2.44949 & 0 & 2.64575 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2.64575 & 0 & 2.82843 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.82843 & 0 \end{pmatrix} \tag{21}$$

The resultant approximate and actual transformations are given in Figure (11). The actual transformation was found using Laplace Transforms on the differential equation

$$\ddot{x} + \frac{c}{m}\dot{x} + \frac{k}{m}x = 0. \tag{22}$$

The exact transformation is

$$u(T,\xi) = \frac{2ve^{-\frac{Tc}{2m}}}{\sqrt{\frac{c}{m}^2 - \frac{4}{m}(\bar{k} + \xi\tilde{k})}}\ sinh\left[\frac{T}{2}\sqrt{(\frac{c}{m})^2 - \frac{4}{m}(\bar{k} + \xi\hat{k})}\right]. \tag{23}$$

The error plot shown in Figure (12), calculated as $|g(\xi) - \hat{g}(\xi)|$, verifies that the $\hat{g}(\xi)$ produced with polynomial chaos of degree 8 is a good approximation to the actual transformation. The error decreases as the degree, of the Hermite polynomial used in the expansion, increases. The Mathematica code used for the Hermite-chaos method is found in Appendix $A.1$.

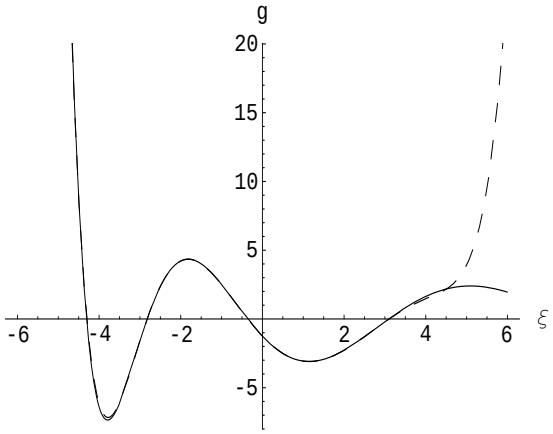


Figure 11: Approximate (dashed) and Actual (solid) Transformation of $Y = g(\xi)$ Using Hermite-chaos of Degree 8.

While Hermite-Chaos was shown to be an effective method of approximating the transformation for the linear and nonlinear ODE examples, polynomial chaos, in general, can fail when the transformation $g : X \to Y$ is not smooth. For ex-

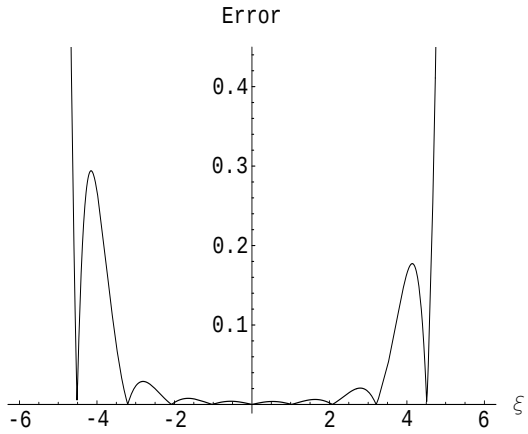


Figure 12: Error Plot $Y = |g(\xi) - \hat{g}(\xi)|$ Using Hermite-chaos of Degree 8.

ample, the transformation $g(X) = tan(X)$, where X is a random variable from the $Uniform(0, \pi)$ distribution, is discontinuous at $x = \frac{\pi}{2}$. When using polynomial chaos, with Legendre polynomials as the basis functions determined from Table (1), the results, in Figure (13), are what we expect, but do not approximate the transformation well. By using the uniform sampling method, described in the next section, for $g(X) = tan(X)$, Figure (13) shows that the uniform sampling method approximates the true transformation quite well.

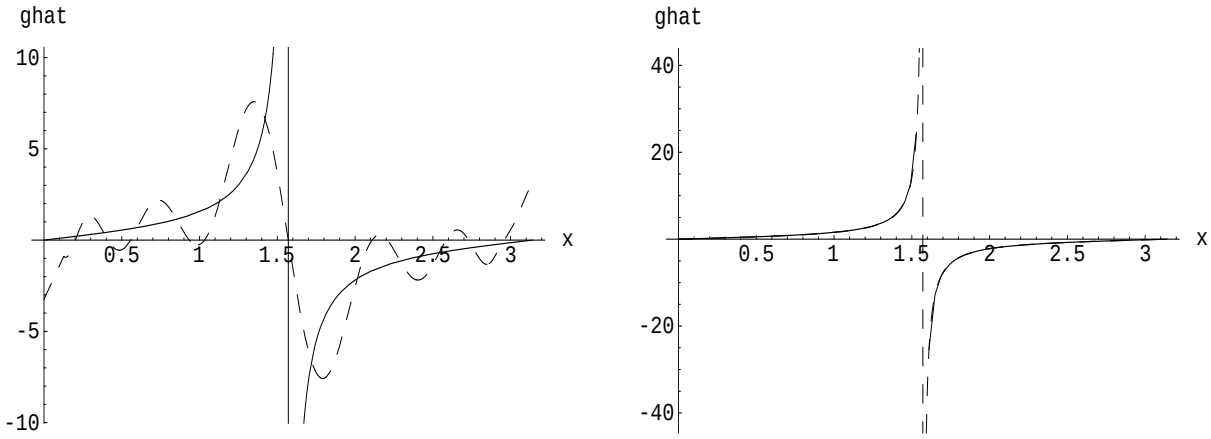


Figure 13: Approximate (dashed) and Actual (solid) Transformation of $Y = tan(X)$ Using Polynomial Chaos of Degree 8 and Uniform Sampling Method With 80 Samples Where X Uniform(0,π)

3.1.2 ***Approximating the Transformation*** $g : X \to Y$ ***Using the Uniform Sampling Method.*** The second method to approximate the transformation $y = g(X)$ is by uniformly sampling the transformation and using interpolation. Approximate the transformation $g(x)$ using a set $\{(x_i, \hat{g}(x_i)\}$. In this case, the set $\{x_1, \ldots, x_n\}$ is not random but is chosen to be evenly spaced on the interval $[a, b]$. Then, $\{\hat{g}(x_1), \ldots, \hat{g}(x_n)\}$ is calculated by solving the ode system for each x_i. Using this data, an approximation $\hat{g}(x) \approx g(x)$ is constructed. The approximated and actual transformation are shown in Figure (14) where $g(X)$ is the solid plot and $\hat{g}(X)$ is the dashed plot. The Mathematica code used for the uniform sampling method can be found in Appendix *A*.2.

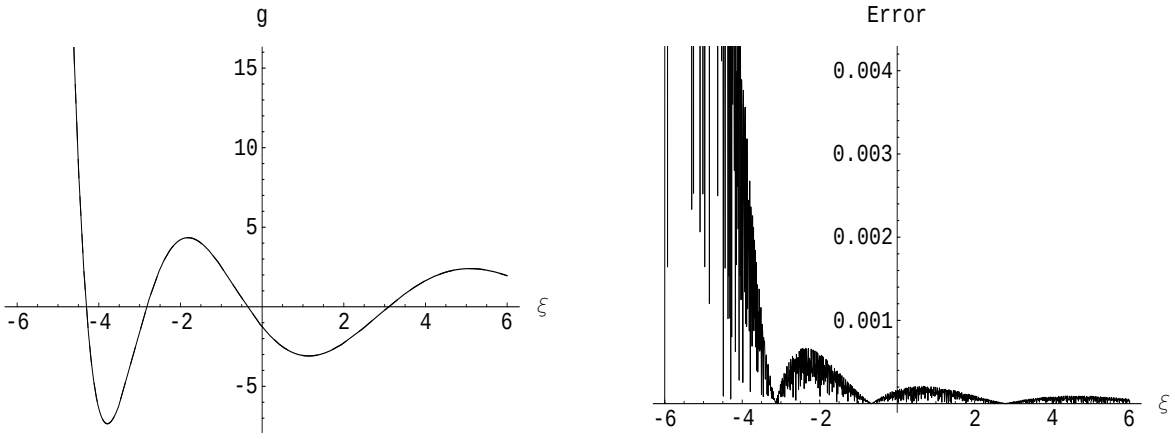


Figure 14: Actual and approximate Transformation $Y = g(X)$ Using Uniform Sampling Method With 500 Samples and Error Plot $Y = |g(X) - \hat{g}(X)|$

3.1.3 ***Finding the Distribution With Approximate Transformation.*** Once the approximation of the transformation $y = g(X)$ is found, the distribution of the output variable can be obtained. The methods employed to accomplish this are Monte Carlo and the transformation method both described previously in Table (2).

3.1.3.1 ***Monte Carlo Method.*** The distribution of the output variable can be determined using the "input" Monte Carlo method. This is a technique

for approximating the pdf using a histogram generated by producing random inputs and solving the system for each random input. The result is obtained in Figure (15).

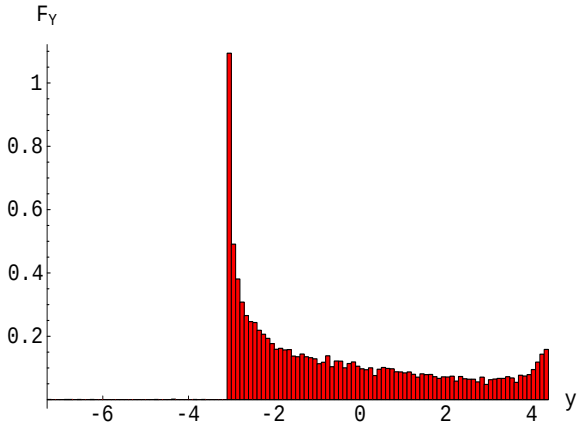


Figure 15: PDF of Y Using "Input" Monte Carlo With 20,000 Runs

The distribution of the output variable can also be determined using the "output" Monte Carlo method. This is a technique for approximating the pdf by approximating the transformation and use Monte Carlo with $\hat{g}(X)$ evaluating the random inputs. The result is shown in Figure (16). The Mathematica code for Monte Carlo simulation can be found in Appendix D.

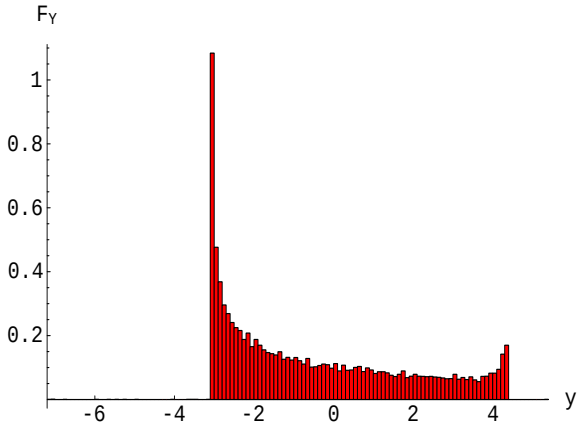


Figure 16: PDF of Y Using "Output" Monte Carlo With 20,000 Runs Where $\hat{g}(x)$ Obtained by Hermite-Chaos of Degree 8.

3.1.3.2 ***Transformation Method.*** Using $\hat{g}(x)$, the distribution of the output variable can also be determined using the transformation method. First, a sample $X = \{x_1, \ldots, x_n\}$ is generated. This sample is random for the stochastic projection method and equally spaced over the interval $[a, b]$ for the uniform sampling method. Then, we evaluate $Y = \hat{g}(x) = \{y_1, \ldots, y_n\}$ producing the sample x_i, y_i. Next, we go through the list of increasing y values, finding the limits of integration and integrating to generate the data to build the cumulative distribution function (cdf). Then the cdf is differentiated to get the pdf. Both the cdf and pdf are shown in Figure (17). The Mathematica code for the transformation method can be found in Appendix C.

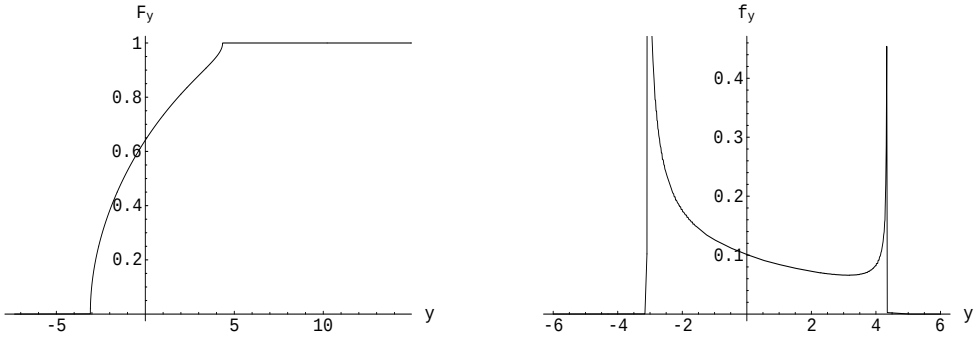


Figure 17: CDF and PDF of Y Using Transformation Method With $\hat{g}(X)$ Obtained by Uniform Sampling With 512 Samples

The methods used to obtain the distribution of the output random variable of a system with one random input are outlined in Table (2). The approximated transformation $\hat{g}(X)$ was found for the linear ODE with the uniform sampling method and polynomial chaos. In chapter 4, the accuracy and efficiency of the uniform sampling method and the Hermite-chaos method will be analyzed by comparing the resulting approximate transformations to the true transformation. In addition, we will use Monte Carlo simulation and the transformation method to obtain the pdfs using the approximated transformations determined from Hermite-chaos and uniform sampling. These pdfs will also be compared to the actual pdfs to determine the accuracy and efficiency of these methods.

3.2 Nonlinear Example

In this section, we consider a nonlinear problem. The equation of the spring mass system is

$$\begin{aligned} &m\ddot{u} + k_1\dot{u} + k_2 u + (\bar{k_3} + \xi\tilde{k_3})u^3 = 0 \\ &u(0) = w \\ &u'(0) = v \end{aligned} \tag{24}$$

which is similar to the linear case except now there is a nonlinear cubic spring stiffness. Since $g(X)$ is not known in this case, it must be approximated. The approximation to the transformation will be obtained by the stochastic projection method and the uniform sampling method. Then, the transformation method or Monte Carlo Simulation will be used to obtain the distribution of the output random variable. The following computations were carried out with ξ being a standard normal random variable and the following parameters for the model problem (24). The actual transformation, that will be used for comparison, was obtained by sampling using 8192 samples. The result is shown in Figure (18).

t	m	k_1	k_2	$\bar{k_3}$	$\tilde{k_3}$	w	v
10	10	0	10	1	0.02	10	0

Table 4: **Parameters for Nonlinear Model ODE (24)**

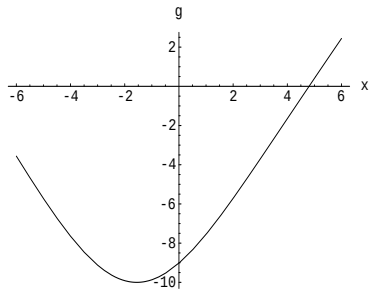


Figure 18: Actual Transformation by Uniform Sampling Method Using 8192 Samples

3.2.1 ***Approximating the Transformation*** $g : X \to Y$ ***Using the Stochastic Projection Method.*** This method follows the same idea given for the linear ODE described previously. The method is shown by the following formulations.

Consider the model problem from (24). The approximate transformation is given by

$$u(t,\xi) = \sum_{i=0}^{P} x_i(t)\phi_i(\xi) \tag{25}$$

and we define k to be

$$k = \bar{k_3} + \xi\tilde{k_3} \;\; where\, \xi \sim N(0,1). \tag{26}$$

Next, we substitute Equations (26) and (25) into (24). This results in

$$\sum_{i=0}^{P} \ddot{x}_i(t)\phi_i + \frac{k_2}{m}\sum_{i=0}^{P} x_i(t)\phi_i + \frac{(\bar{k_3} + \xi\tilde{k_3})}{m}(\sum_{i=0}^{P} x_i(t)\phi_i)^3 = 0. \tag{27}$$

Then we multiply by ϕ_j, where $j = 0, \ldots, P$, and a weight function $w(\xi)$ and integrate on $[a, b]$. Letting $< f, g > \; = \int_a^b f(\xi)g(\xi)w(\xi)\, d\xi$, we get

$$
\begin{aligned}
&\textstyle\sum_{i=0}^{P} \ddot{x_i}(t) < \phi_i, \phi_j > \; + \frac{k_2}{m} \sum_{i=0}^{P} x_i(t) < \phi_i, \phi_j > + \\
&\bar{\tfrac{k_3}{m}} < \textstyle\sum_{i=0}^{P} x_i(t)\phi_i(\xi) \sum_{k=0}^{P} x_k(t)\phi_k(\xi) \sum_{l=0}^{P} x_l(t)\phi_l(\xi), \phi_j > + \\
&\tilde{\tfrac{k_3}{m}} < \xi \textstyle\sum_{i=0}^{P} x_i(t)\phi_i(\xi) \sum_{k=0}^{P} x_k(t)\phi_k(\xi) \sum_{l=0}^{P} x_l(t)\phi_l(\xi), \phi_j >= 0 \\
&for \text{ j=0,...,P.}
\end{aligned}
\tag{28}
$$

This equation can be written as

$$
\begin{aligned}
&\textstyle\sum_{i=0}^{N} (\ddot{x_i}(t) + \frac{k_2}{m} x_i(t)) < \phi_i, \phi_j > \; + \\
&\bar{\tfrac{k_3}{m}} \textstyle\sum_{i=0}^{P} \sum_{k=0}^{P} \sum_{l=0}^{P} x_i(t)x_k(t)x_l(t) < \phi_i\phi_k\phi_l, \phi_j > + \\
&\tilde{\tfrac{k_3}{m}} \textstyle\sum_{i=0}^{P} \sum_{k=0}^{P} \sum_{l=0}^{P} x_i(t)x_k(t)x_l(t) < \xi\phi_i\phi_k\phi_l, \phi_j >= 0 \\
&for \text{ j=0,...,P.}
\end{aligned}
\tag{29}
$$

This system of j equations along with the initial conditions needs to be solved to find $x_k(t)$ in Equation (25). Once this is determined, the transformation can be approximated the same way as in the linear case.

From the parameters given for the model ODE in Table (4), we determine the approximate transformation for the nonlinear problem. For this example, the stochastic projection method described uses the one-dimensional Hermite polynomials in terms of normally distributed random variables (ξ) with a mean of 0 and variance of 1 as described in [4].

For the function $\phi(\xi, i)$, we used the scaled Hermite polynomial

$$
\phi(\xi, i) = \frac{H_i[\frac{\xi}{\sqrt{2}}]}{\sqrt{2^i i!}} \tag{30}
$$

and for the weight function $w(\xi)$, we use the pdf of the standard normal distribution

$$
w(\xi) = \frac{1}{\sqrt{2\pi}} e^{-\frac{\xi^2}{2}} \tag{31}
$$

The limits of integration used for the inner product calculations are $(-\infty, \infty)$ and $i, j = 0, \ldots, P$. The approximated and actual transformations are shown in Figure (19) where $g(\xi)$ is the solid plot and $\hat{g}(\xi)$ is the dashed plot. The Mathematica code for the Hermite-chaos method can be found in Appendix B.1.

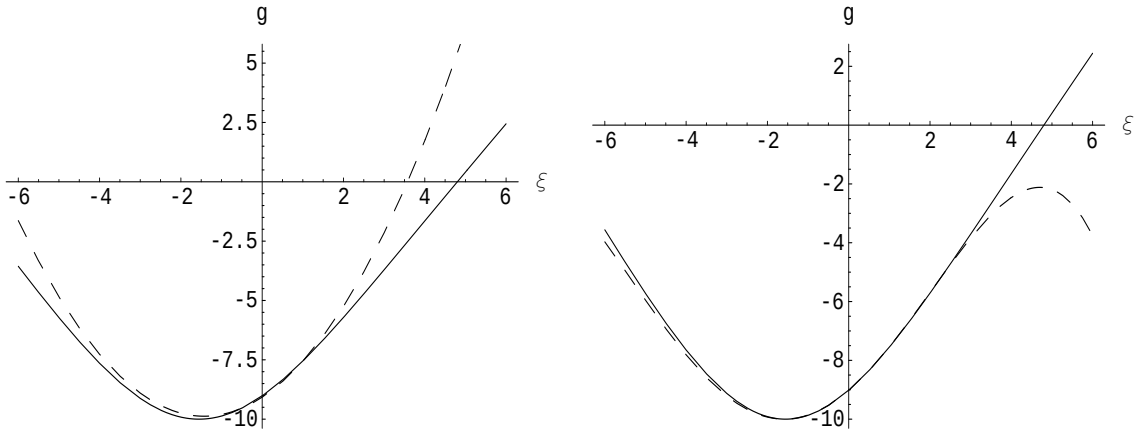


Figure 19: Nonlinear ODE Transformation Using Hermite-chaos of Degree 2 and 4.

3.2.2 ***Approximating the Transformation*** $g : X \to Y$ ***Using the Uniform Sampling Method.*** The uniform sampling method used for the nonlinear case is exactly the same as the linear case. We approximate the transformation $g(x)$ using a set $\{(x_i, g(x_i)\}$ where $\{x_1, \ldots, x_n\}$ is not random but is chosen to be evenly spaced on the interval $[a, b]$. Then, $\{\hat{g}(x_1), \ldots, \hat{g}(x_n)\}$ is calculated by solving the ode system for each x_i. Using this data, an approximation $\hat{g}(x) \approx g(x)$ is constructed. The approximated transformation is shown in Figure (20) where the actual transformation is the solid plot and the approximated transformation is the dashed plot. These two plots are almost identical. The Mathematica code for the uniform sampling method can be found in Appendix B.2. Once we have $\hat{g}(X)$, the transformation method can be used to obtain the cdf. Then by differentiating the cdf, the pdf, in Figure (21), is found. This pdf obtained with the uniform sampling method can be compared to the pdf found using Monte Carlo Simulation. The Monte Carlo pdf is also shown in Figure (21).

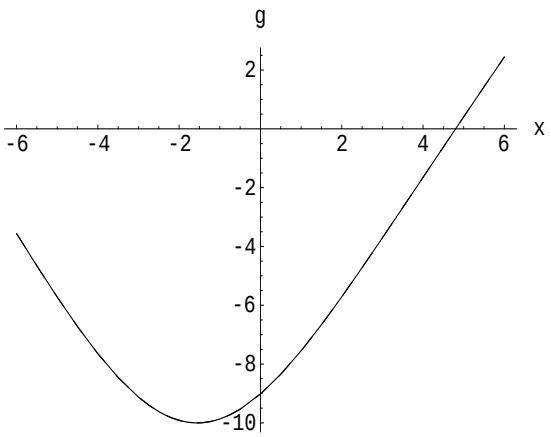


Figure 20: Nonlinear ODE Transformation Using Uniform Sampling Method With 256 Samples

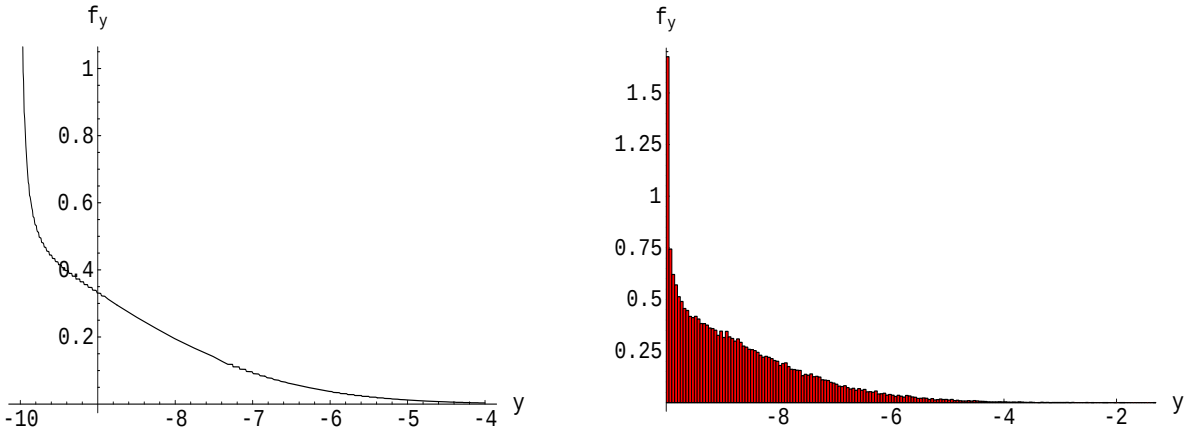


Figure 21: PDF of Nonlinear ODE Using Uniform Sampling Method With 256 Samples and Monte Carlo Simulation With 50,000 Runs

IV. Comparison of Hermite-chaos and Uniform Sampling Method Used to Approximate the Transformation

As demonstrated previously, Hermite-chaos and the uniform sampling method both worked well to approximate the transformation for linear and nonlinear ODE's when the functional form of $g(X)$ was not explicitly known. This chapter will provide a comparison of these methods based on the accuracy of the approximation at different levels and the amount of computational cost required for each method.

4.1 Accuracy and Computational Cost Evaluation of Approximated Transformation

The determination of how accurate the approximations are will be made by computing the weighted error for different levels of each approximation method. The method used to assess the accuracy is

$$E_{\hat{T}} = \sqrt{\int_{-6}^{6} (T(\xi) - \hat{T}(\xi))^2 w(\xi) dx} \tag{32}$$

where $T(\xi)$ is the actual transformation, $\hat{T}(\xi)$ is the approximate transformation, and $w(\xi)$ is the pdf of the standard normal, $N(0,1)$, distribution since we assume the input random variable can be written in terms of the standard normal random variable. The Mathematica code for the weighted error calculations can be found in Appendix A. The computational cost was measured by the time, in seconds, each method took to execute using Mathematica. Only the amount of computer code necessary to obtain the approximated transformation was processed. The TimeUsed function in Mathematica recorded the time spent in the kernel to execute the selected code. This is the method for recording the computational time that will be used throughout this paper.

4.1.1 ***Linear ODE.*** The linear ODE under analysis is the model problem (1) with the parameters in Table (4) from Chapter 3.

4.1.1.1 ***Hermite-chaos.*** To evaluate the error, when using Hermite-chaos to find the approximation, the weighted error was calculated using Hermite polynomials with degree 2, 4, 6, 8, 10, and 12. As the degree of the Hermite polynomial increases, the accuracy of $\hat{g}(X)$ increases dramatically. However, the computational cost also increases significantly. The graphs in Figure (22) show the increase in accuracy between hermite polynomial of degree 2 and 4 respectively.

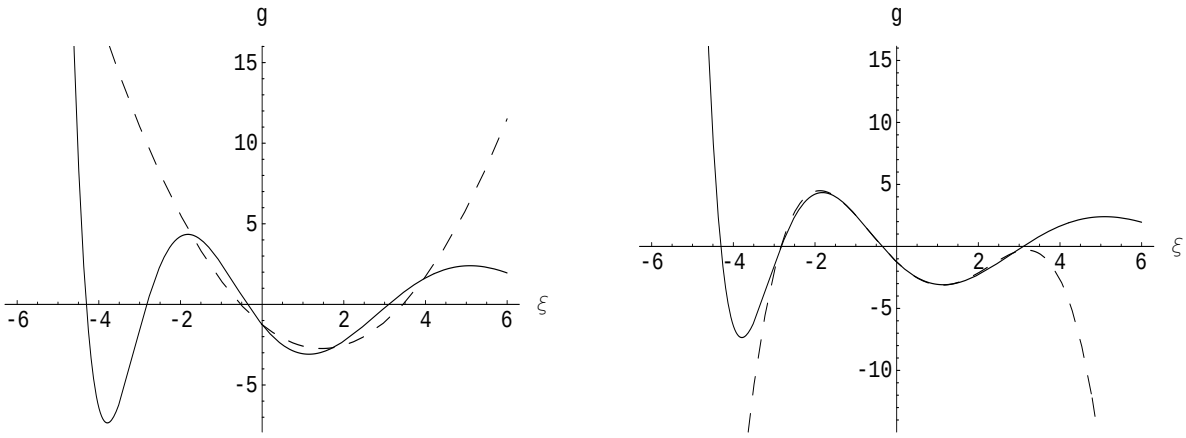


Figure 22: Approximate (dashed) and Actual (solid) Transformation of $Y = g(X)$ Using Hermite-chaos of Degree 2 and 4 for Linear ODE.

The weighted error for degree 2 and 4 are 1.10399 and 0.306322 respectively. The error continues to decrease for each increase in even degree. The log-log graph in Figure (23) shows the weighted error for each degree of the hermite polynomial analyzed. We see that the error decrease is exponential.

4.1.1.2 ***Uniform Sampling Method.*** For this method, the weighted error was calculated using increasing number of samples (8, 16, 32, 64, 128, 256, 512, 1024, 2048, and 4096). These are simply determined by letting the sample size be 2^n where $n = 3, 4, 5 \ldots 12$. As the number of samples increases, the accuracy of $\hat{g}(X)$ also increases. The graphs in Figure (24) show the increase in accuracy between samples

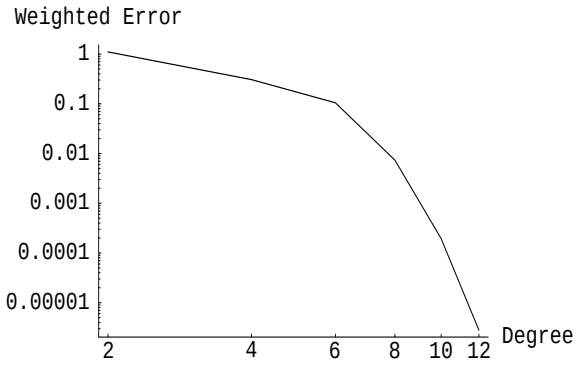


Figure 23: Log-Log Weighted Error Plot by Degree of Hermite Polynomial for Linear ODE.

of 32 and 64 respectively. The weighted error for 32 and 64 samples are 0.0389027 and 0.00973277 respectively. The error continues to decrease for each increase in number of samples. The log-log graph in Figure (25) shows the weighted error for each sample set analyzed.

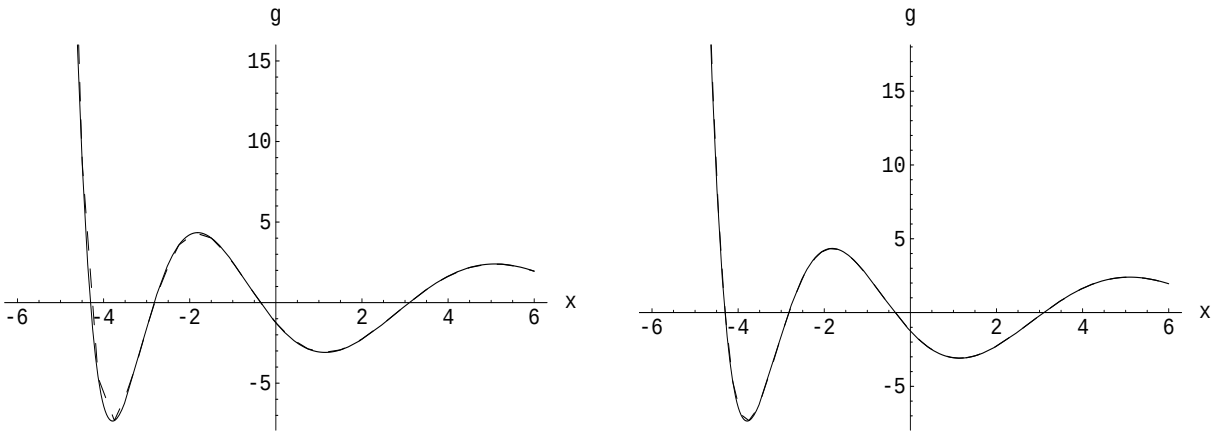


Figure 24: Approximate (dashed) and Actual (solid) Transformation of $Y = g(X)$ Using Uniform Sampling Method With 32 and 64 Samples for Linear ODE.

In Table (5), the accuracy and computational time of the uniform sampling method and Hermite-chaos are compared. We see that the uniform sampling method produces better results more efficiently. When comparing the Hermite-chaos method

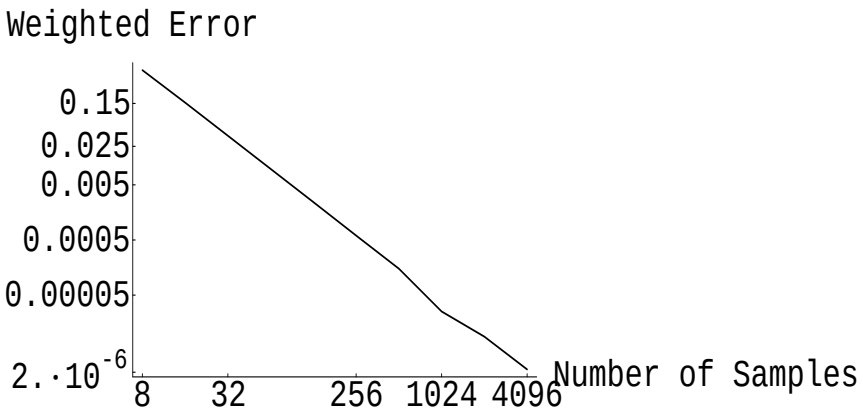


Figure 25: Log-Log Weighted Error Plot by Number of Samples for Linear ODE.

of degree 4 and the uniform sampling method with 64 samples, we see that the uniform sampling method produces a more accurate result in roughly the same amount of computational time. This is also the case for Hermite-chaos of degree 8 and 256 samples. This difference disappears as polynomial degree and number of samples increase.

	Hermite-Chaos			**Uniform Sampling**	
Degree	**Time(Seconds)**	**Error**	**Samples**	**Time(Seconds)**	**Error**
2	0.733	1.10399	32	.687	0.0389027
4	0.876	0.306322	64	.860	0.00973277
6	1.219	0.104656	128	1.11	0.00243358
8	1.656	0.007349	256	1.671	0.000601922
10	2.376	0.0001943	512	2.751	0.000150971

Table 5: **Comparison of Computational Time and Error Between Hermite-Chaos and Uniform Sampling Method Used to Approximate the Transformation for the Linear ODE**

When analyzing the error rate of convergence, for the uniform sampling method, we see that the error converges at a quadratic rate which is consistent with linear interpolation methods. The error rate of convergence (ROC) for the linear case was calculated using the following formula

$$ROC = \frac{log(\frac{Error(2048)}{Error(4096)})}{log(\frac{4096}{2048})} \tag{33}$$

The weighted errors for 2048 and 4096 samples are 8.8297x10^{-6} and 2.2466x10^{-6} respectively. The error rate of convergence was calculated to be 1.97464.

4.1.2 ***Nonlinear ODE.*** The procedures for assessing the weighted error of the nonlinear ODE are exactly the same as the linear ODE. The "actual transformation" was obtained by the sampling method with 8192 samples. This will be used to compare the accuracy of the different levels of each approximation method. The Mathematica code for the weighted error calculations can be found in Appendix B. The following results were obtained.

4.1.2.1 ***Hermite-chaos.*** Again, as the degree of the hermite polynomial increased, the accuracy of $\hat{g}(X)$ increased dramatically. The graphs in Figure (26) show the increase in accuracy between hermite polynomial of degree 2 and 4 respectively. The weighted error for degree 2 and 4 are 0.167758 and 0.0164006 respectively. The error continues to decrease for each increase in even degree. The log-log graph in Figure (27) shows the weighted error for each degree of the hermite polynomial analyzed. Again, we see that the error decrease is exponential.

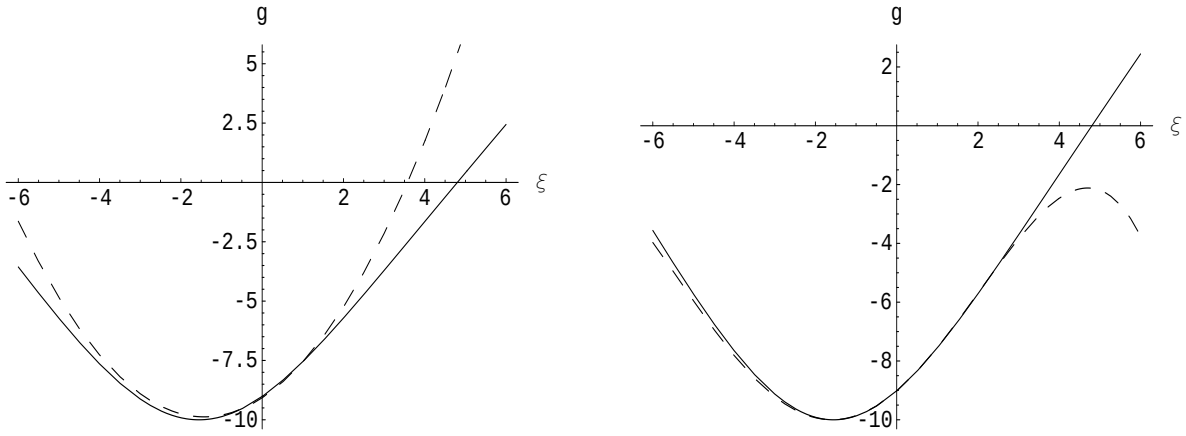


Figure 26: Approximate (dashed) and Actual (solid) Transformation of $Y = g(X)$ Using Hermite-chaos of Degree 2 and 4 for Nonlinear ODE.

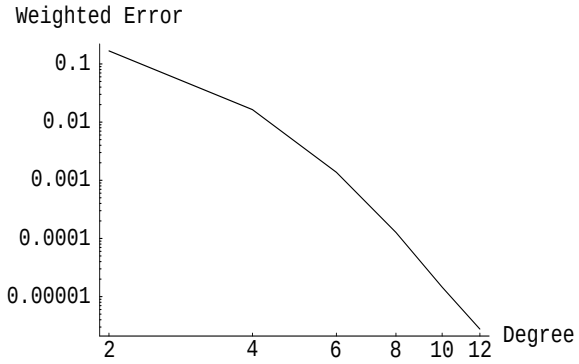


Figure 27: Log-Log Weighted Error Plot by Degree of Hermite Polynomial for Nonlinear ODE.

4.1.2.2 ***Uniform Sampling Method.*** In the nonlinear case, just as the linear, when the number of samples increases the accuracy of $\hat{g}(X)$ increases. The graphs in Figure (28) show the increase in accuracy between samples of 32 and 64. The weighted error for 32 and 64 samples are 0.00802643 and 0.00200724 respectively. The error continues to decrease for each increase in number of samples. The graph in Figure (29) shows the weighted error for each sample set analyzed.

In Table (6), the accuracy and computational time of the uniform sampling method and Hermite-chaos are compared for this nonlinear example. We see that

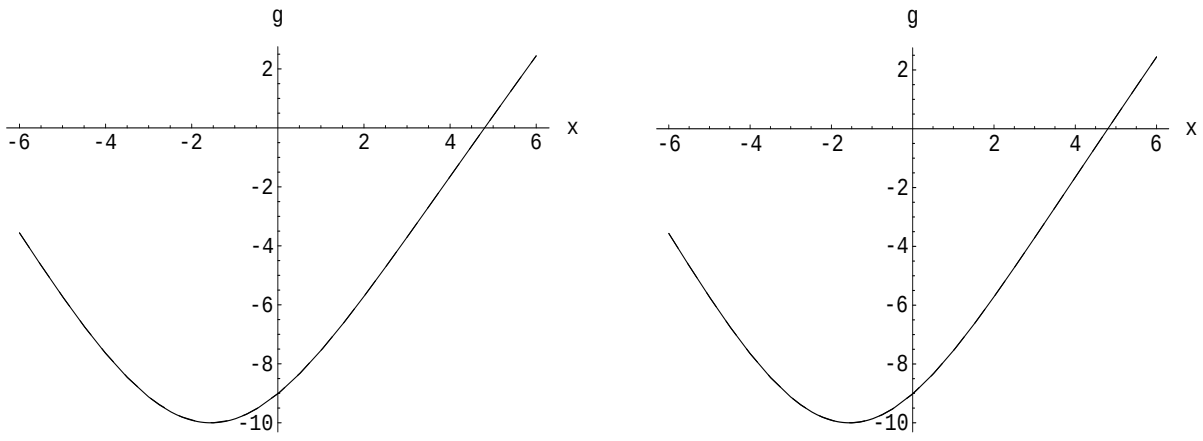


Figure 28: Approximate (dashed) and Actual (solid) Transformation of $Y = g(X)$ Using Uniform Sampling Method With 32 and 64 Samples for Nonlinear ODE.

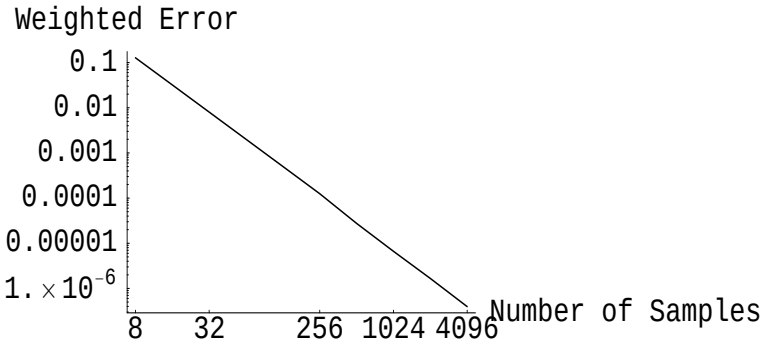


Figure 29: Log-Log Weighted Error Plot by Number of Samples for Nonlinear ODE.

the uniform sampling method produces better results more efficiently. When comparing the Hermite-chaos method of degree 4 and the uniform sampling method with 32 samples, we see that the uniform sampling method produces a more accurate result in much less computational time. There is a more dramatic difference between the Hermite-chaos of degree 8 and 256 samples. While the error is approximately the same, the Hermite-chaos method to approximate the transformation takes close

to 840 times longer than the sampling method. In contrast to the linear example described previously, the uniform sampling method outperforms the Hermite-chaos method significantly for the nonlinear example. The computational time required to run the Hermite-chaos method is mostly spent on the calculation of the inner products in Equation (29). When these inner products were precalculated and the results were substituted into the Mathematica code, the computational cost was greatly reduced. Once the needed inner products are calculated, they can be substituted in for any degree of Hermite polynomial needed for finding $\hat{g}(X)$ from the model problem in (24). The limitation on this procedure is that the inner products have to be recalculated for different types of nonlinearities. The results using the precalculated inner products are shown in Table (7).

	Hermite-Chaos			**Sampling Method**	
Degree	**Time(Seconds)**	**Error**	**Samples**	**Time(Seconds)**	**Error**
2	12.296	0.1677582	32	0.780	0.00802643
4	136.86	0.0164006	64	1.11	0.00200724
6	632.20	0.0013722	128	1.609	0.000501754
8	2265.7	0.0001269	256	2.688	0.000125043
10	7114.8	0.0000145	512	4.891	0.0000275471

Table 6: **Comparison of Computational Time and Error Between Hermite-Chaos and Uniform Sampling Method Used to Approximate the Transformation for the Nonlinear ODE**

	Hermite-Chaos	
Degree	**Time(Seconds)**	**Error**
2	1.736	0.16776
4	1.797	0.01640
6	1.969	0.001372
8	2.282	0.000127
10	3	0.0000145

Table 7: **Computational Time and Error of Hermite-Chaos Method With Precalculated Inner Products for the Nonlinear ODE**

When analyzing the error rate of convergence, for the uniform sampling method, we see that the error converges at a quadratic rate which is consistent with linear interpolation methods. The error rate of convergence for the nonlinear case was calculated using the following formula

$$ROC = \frac{log(\frac{Error(2048)}{Error(4096)})}{log(\frac{4096}{2048})} \tag{34}$$

The weighted errors for 2048 and 4096 samples are $1.6645x10^{-6}$ and $3.9598x10^{-7}$ respectively. The error rate of convergence was calculated to be 2.07156.

4.2 Accuracy Evaluation of Approximated PDF

Now that the accuracy of $\hat{g}$, approximated with Hermite-chaos and the uniform sampling method, has been analyzed, we will look at the precision of the pdf generated from these approximated transformations. As discussed in Chapter 2, there were two methods of approximating the pdf; the transformation method and Monte Carlo simulation. The approximated pdfs will be compared to the pdfs obtained from the actual transformations for the linear and nonlinear examples. The method used to assess the accuracy is

$$E_{\hat{T}} = \sqrt{\int_a^b (pdf(y) - \hat{pdf}(y))^2 dy} \tag{35}$$

where $pdf(y)$ is the actual pdf and $\hat{pdf}(y)$ is the approximated pdf. The interval for the linear example will be $(a, b) = (-3, 4.1)$ while the interval for the nonlinear example will be $(a, b) = (-10, -4)$. These modified intervals are necessary due to erratic behavior at the edges of the pdf's from the use of the interpolation function in Mathematica. The limits selected adequately capture where the actual pdf has positive probability.

4.2.1 ***Linear ODE.***

4.2.1.1 ***Transformation Method.*** To evaluate the error when finding the pdf with the transformation method using Hermite-chaos, the error was calculated using the actual pdf and the approximated pdf using hermite polynomials with degree 2, 4, 6, 8, 10, and 12. The graphs in Figure (30) show the increase in accuracy between hermite polynomials of degree 2 and 4. The error for degree 2 and 4 are 0.0228001 and 0.0154914 respectively. The error continues to decrease for each increase in even degree. The graph in Figure (31) shows the error for each degree of the hermite polynomial analyzed.

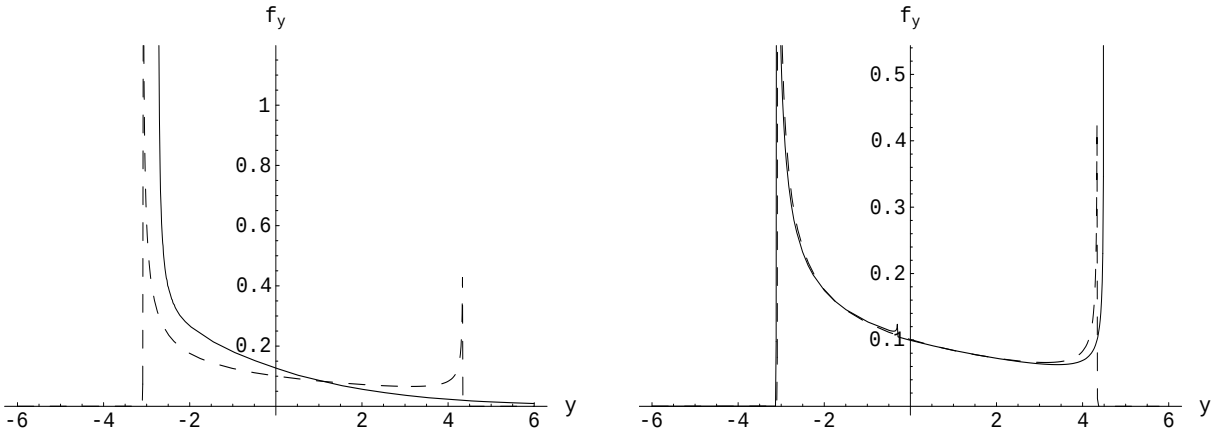


Figure 30: Approximate (solid) and Actual (dashed) $pdf(X)$ (Via Transformation Method) Using Hermite-chaos of Degree 2 and 4 for the Linear ODE.

To evaluate the error when finding the pdf with the transformation method using uniform sampling, the error was calculated using the actual pdf and the approximate pdf with sample sizes of 2^n where $n = 4, 5, \ldots 12$. The sample size of 8 was too small to compute a pdf via the transformation method. The graphs in Figure (32) show the increase in accuracy between samples of 32 and 64. The error for 32 and 64 samples are 0.0640926 and 0.0178453 respectively. The error continues to decrease for each increase in number of samples. The graph in Figure (33) shows the error for each sample set analyzed. The comparison between Hermite-chaos and the uniform sampling method, in terms of pdf error is shown in Table (8).

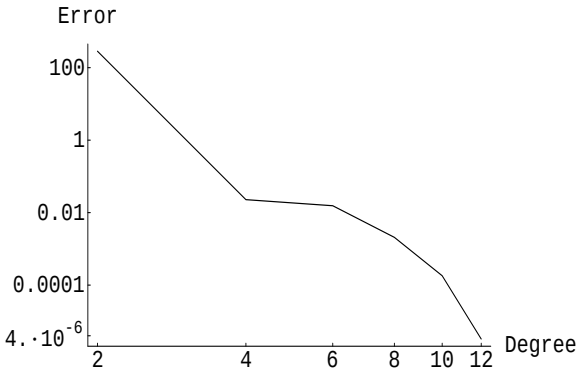


Figure 31: Log-Log Error Plot for $pdf(X)$ (Via Transformation Method) by Degree of Hermite Polynomial for the Linear ODE.

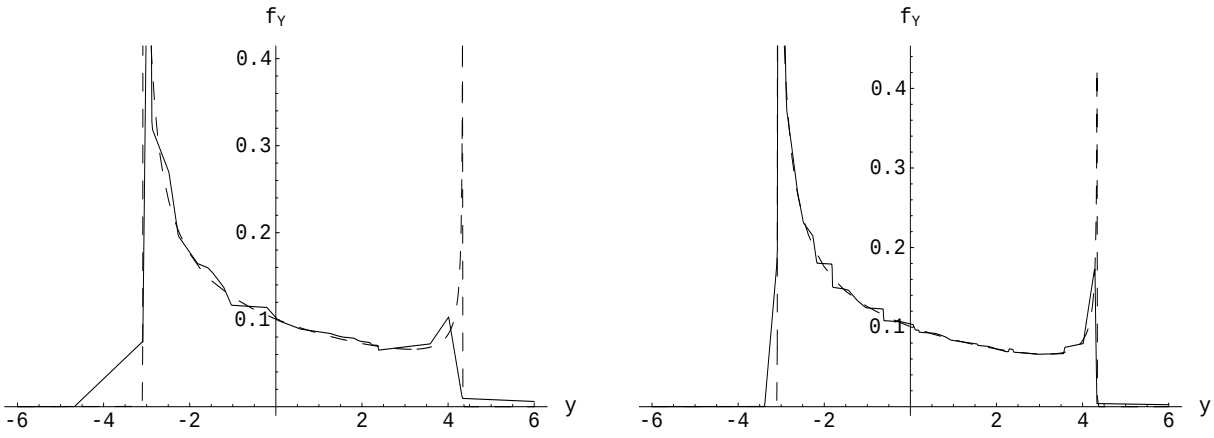


Figure 32: Approximate (solid) and Actual (dashed) $pdf(X)$ (Via Transformation Method) Using Sampling Method With 32 and 64 Samples for the Linear ODE.

4.2.1.2 ***Monte Carlo Method.*** To obtain a pdf from the histogram generated by Monte Carlo simulation, the midpoints of the frequency rectangles were linearly interpolated to form the pdf function to be analyzed. When using Hermite-chaos to approximate the transformation, the error was calculated using the actual pdf and the approximated pdf found using hermite polynomials with degree 2, 4, 6, 8, 10, and 12. The graphs in Figure (34) show the increase in accuracy between hermite polynomial of degree 2 and 4. The error for degree 2 and 4 are 0.423889 and 0.0323146 respectively. The error continues to decrease for each increase in even

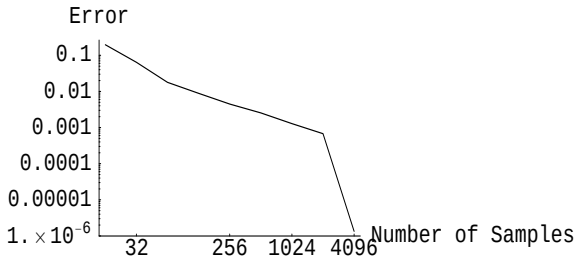


Figure 33: Log-Log Error Plot for $pdf(X)$ (Via Transformation Method) by Number of Samples for the Linear ODE.

Hermite-Chaos		**Sampling Method**	
Degree	**Error**	**Samples**	**Error**
2	285.081	32	0.0640926
4	0.0228001	64	0.0178453
6	0.0154914	128	0.00886257
8	0.00207821	256	0.00595147
10	0.000181367	512	0.00254538
12	.000003249	4096	.000001339

Table 8: **Comparison of Error Between Hermite-chaos and Uniform Sampling Method Used to Approximate the PDF (Via Transformation Method) of the Linear ODE**

degree. The graph in Figure (35) shows the error for each degree of the hermite polynomial analyzed.

To evaluate the error when finding the pdf with Monte Carlo simulation using the uniform sampling method, the error was calculated using the actual pdf and the approximate pdf with sample sizes of 2^n where $n = 3, 5, \ldots 12$. The graphs in Figure (36) show the increase in accuracy between samples of 32 and 64 respectively. The error for 32 and 64 samples are 0.0885416 and 0.024061 respectively. The error fluctuates between each number of samples. This is typical of Monte Carlo simulation

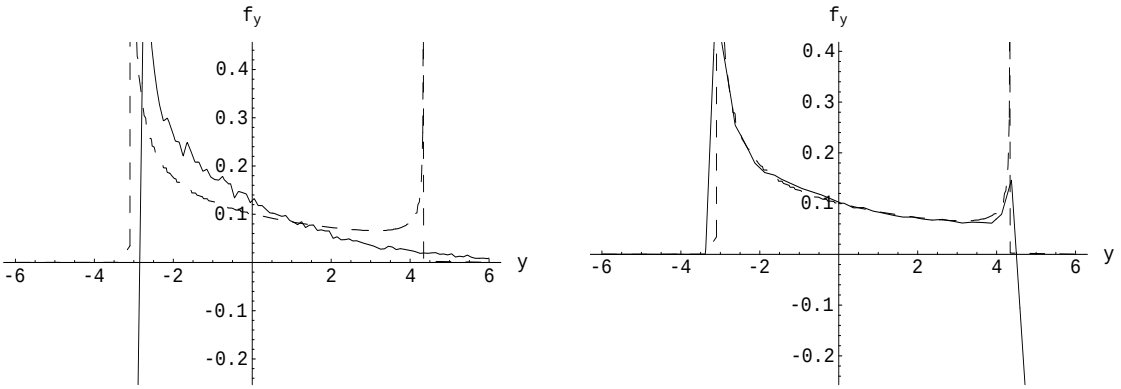


Figure 34: Approximate (solid) and Actual (dashed) $pdf(X)$ (Via Monte Carlo Method) Using Hermite-chaos of Degree 2 and 4 for the Linear ODE.

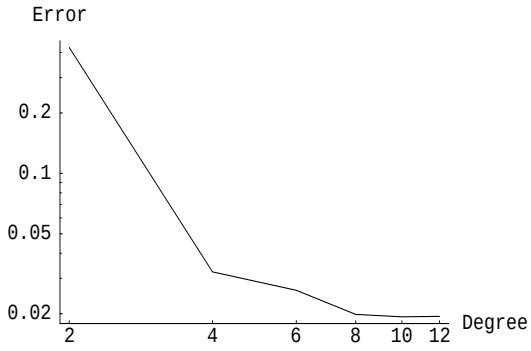


Figure 35: Log-Log Error Plot for $pdf(X)$ (Via Monte Carlo Method) by Degree of Hermite Polynomial for the Linear ODE.

due to the random input used to obtain the pdf. The graph in Figure (37) shows the error for each sample set analyzed.

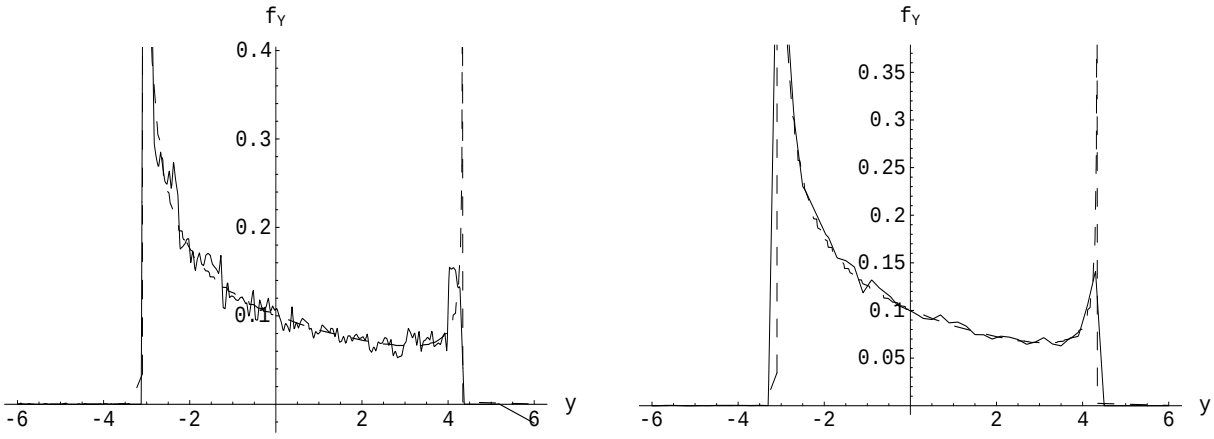


Figure 36: Approximate (solid) and Actual (dashed) $pdf(X)$ (Via Monte Carlo Method)Using Uniform Sampling With 32 and 64 Samples for the Linear ODE.

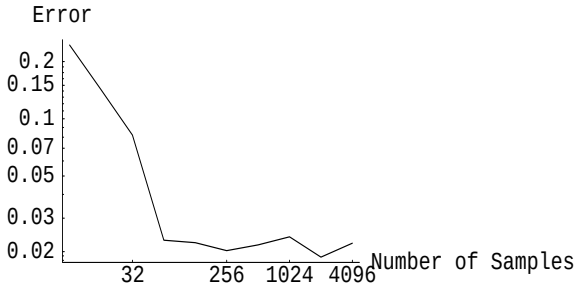


Figure 37: Log Log Error Plot for $pdf(X)$ (Via Monte Carlo Method) by Number of Samples for the Linear ODE.

4.2.2 ***Nonlinear ODE.***

4.2.2.1 ***Transformation Method.*** When using Hermite-chaos, the error was calculated using the actual pdf and the approximated pdf using hermite polynomials with degree 2, 4, 6, 8, 10, and 12. This is the same procedure used for the linear example. The graphs in Figure (42) show the increase in accuracy between hermite polynomial of degree 2 and 4 respectively. The error for degree 2 and 4 are 71.199 and 1.09511 respectively. The error continues to decrease for each increase in

even degree until after degree 6 where the error bottoms out. The graph in Figure (39) shows the error for each degree of the hermite polynomial analyzed.

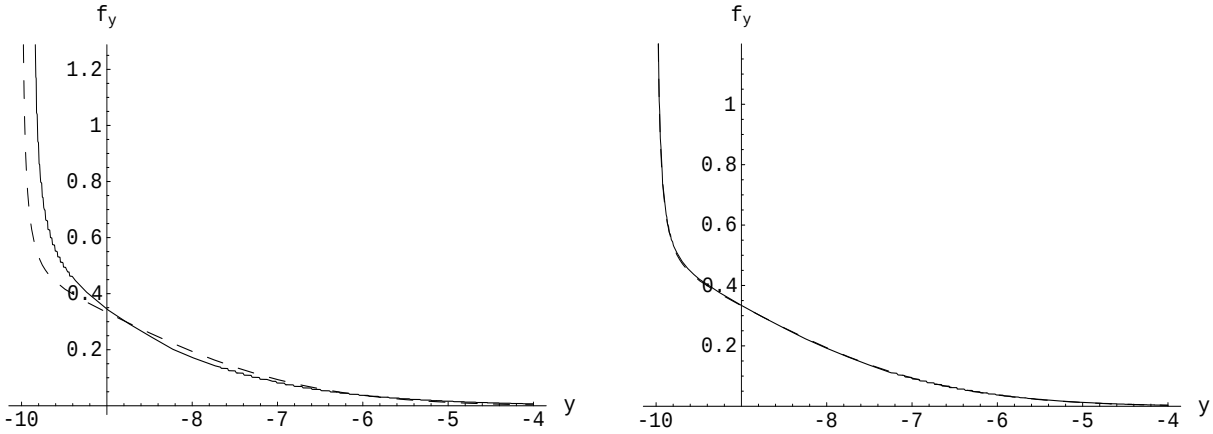


Figure 38: Approximate (Solid) and Actual (dashed) $pdf(X)$ (Via Transformation Method) Using Hermite-chaos of Degree 2 and 4 for Nonlinear ODE.

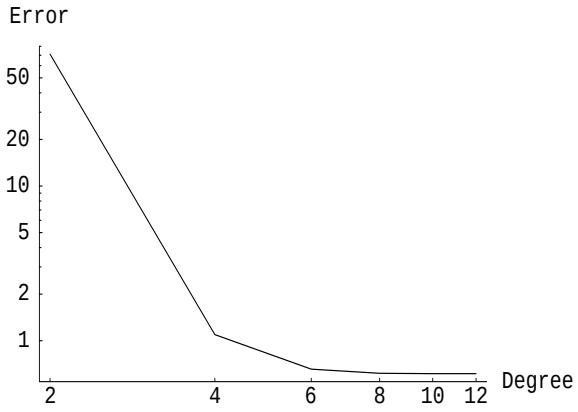


Figure 39: Error Plot for $pdf(X)$ (Via Transformation Method) by Degree of Hermite Polynomial for Nonlinear ODE.

When using uniform sampling, the error was calculated using the actual pdf and the approximated pdf with sample sizes of 2^n where $n = 3, 5, \ldots 12$. The graphs in Figure (40) show the increase in accuracy between samples of 32 and 64 respectively. The error for 32 and 64 samples are 0.723152 and 0.705609 respectively. The error continues to decrease for each increase in number of samples. The graph in Figure (41)

shows the weighted error for each sample set analyzed. The comparison between Hermite-chaos and the sampling method, in terms of pdf error is shown in Table (9).

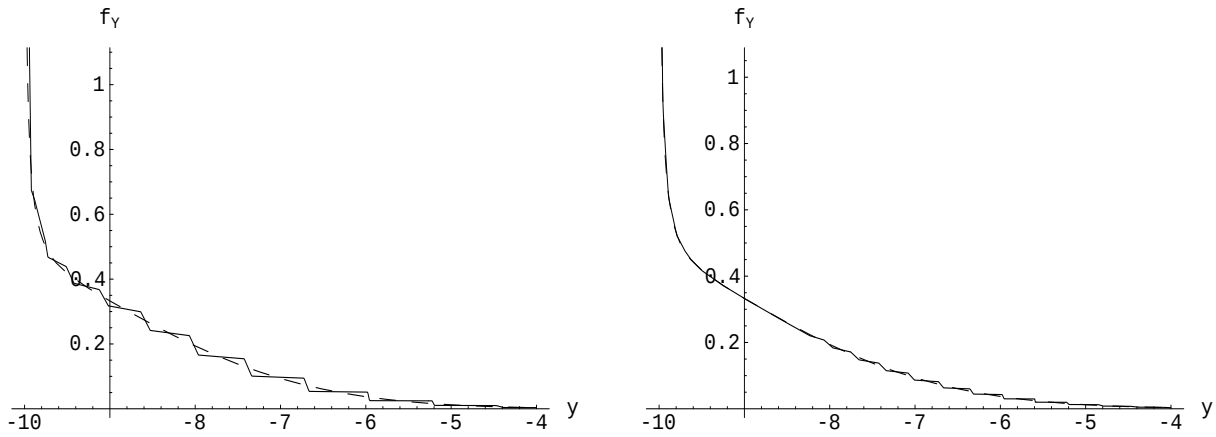


Figure 40: Approximate (solid) and Actual (dashed) $pdf(X)$ (Via Transformation Method) Using Uniform Sampling Method With 32 and 64 Samples for the Nonlinear ODE.

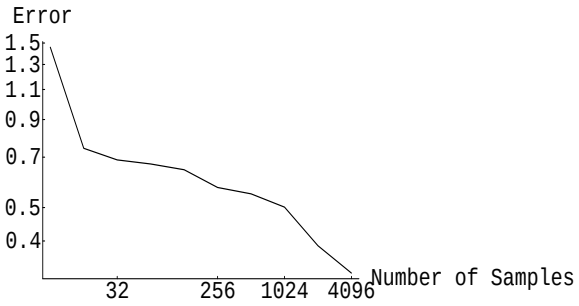


Figure 41: Error Plot for $pdf(X)$ (Via Transformation Method) by Number of Samples for Nonlinear ODE.

4.2.2.2 ***Monte Carlo Method.*** When using Hermite-chaos, the error was calculated using the actual pdf and the approximated pdf using hermite polynomials with degree 2, 4, 6, 8, 10, and 12. This is the same procedure used for the linear example. The graphs in Figure (42) show the increase in accuracy between hermite polynomial of degree 2 and 4. The error for degree 2 and 4 are 0.903489 and

Hermite-Chaos		Sampling Method	
Degree	**Error**	**Samples**	**Error**
2	71.199	32	0.723152
4	1.09511	64	0.705609
6	0.655017	128	0.68144
8	0.615786	256	0.612364
10	0.612495	512	0.589525
12	0.612401	4096	0.380311

Table 9: **Comparison of Error Between Hermite-chaos and Uniform Sampling Method Used to Approximate the PDF (Via Transformation Method) of the Nonlinear ODE**

0.758707 respectively. The error decreases from degree 2 to 4 and fluctuates around 0.758 for degree greater than 4. The graph in Figure (43) shows the error for each degree of the hermite polynomial analyzed.

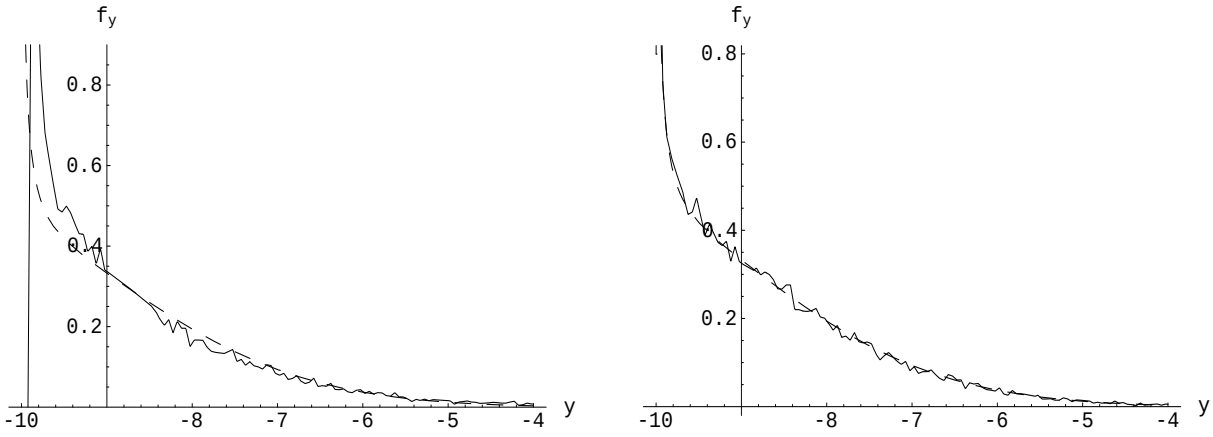


Figure 42: Approximate (solid) and Actual (dashed) $pdf(X)$ (Via Monte Carlo Simulation) Using Hermite-chaos of Degree 2 and 4 for the Nonlinear ODE.

To evaluate the error when finding the pdf with Monte Carlo simulation using the uniform sampling method, the error was calculated using the actual pdf and the approximated pdf with sample sizes of 2^n where $n = 3, 5, \ldots 12$. The graphs in Figure (44), where $pdf(X)$ is the solid plot and $\hat{pdf}(X)$ is the dashed plot, show the increase in accuracy between samples of 32 and 64. The errors for 32 and 64 samples are 0.761875 and 0.758375 respectively. The error decreases for each increase in level

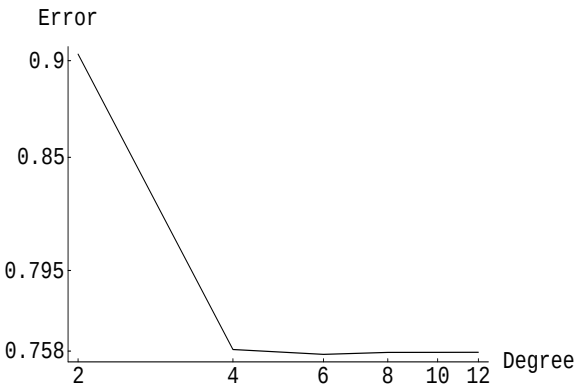


Figure 43: Error Plot for $pdf(X)$ (Via Monte Carlo Simulation) by Degree of Hermite Polynomial for the Nonlinear ODE.

of samples up to 128. From there the error fluctuates above and below 0.757. The graph in Figure (45) shows the error for each sample set analyzed.

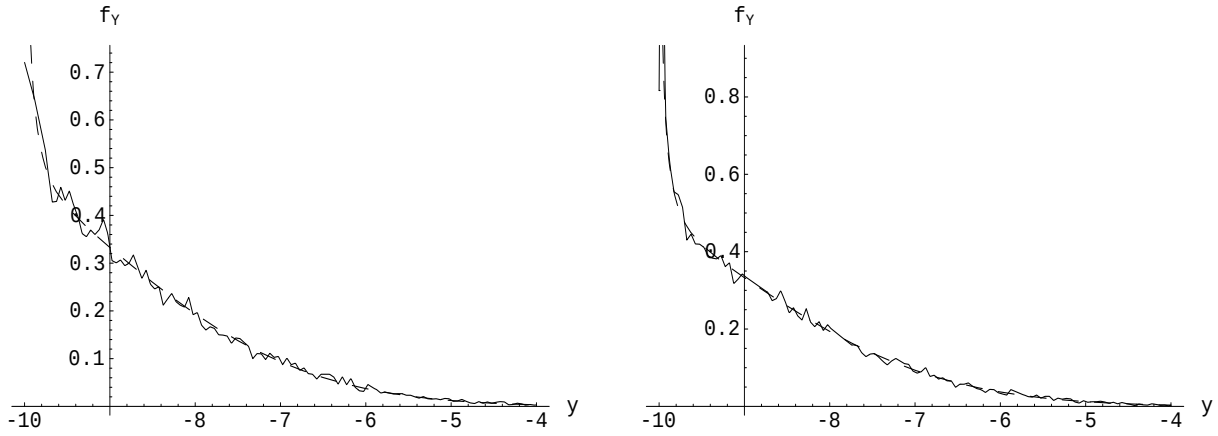


Figure 44: Approximate (solid) and Actual (dashed) $pdf(X)$ (Via Monte Carlo Simulation) Using Uniform Sampling Method With 32 and 64 Samples for the Nonlinear ODE.

It is important to point out that the pdfs generated by Monte Carlo simulation have many oscillations making it difficult to reduce the error. There are methods available that smooth out the pdf, but these are extra operations that need to be performed. When using the transformation method, there are no extra steps involved to

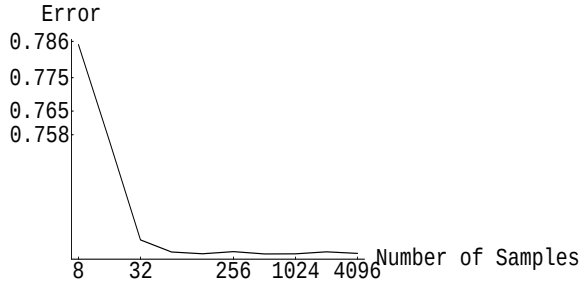


Figure 45: Error Plot for $pdf(X)$ (Via Monte Carlo Simulation) by Number of Samples for the Nonlinear ODE.

get a meaningful pdf function. The transformation method provides a more accurate representation of the pdf generated from the actual or approximated transformation.

V. Nonuniform Sampling Method

In the examples analyzed in the last chapter, uniform sampling was shown to be more accurate and faster than the traditional Monte Carlo and stochastic projection methods currently used in uncertainty analysis. We extend the idea further and consider a nonuniform sampling approach. This idea stems from the notion that the input random variable is from a standard normal distribution. In this case, the values of the random input that are closer to the mean have a higher probability of occurrence than the values that are 2 or 3 standard deviations away from the mean. This implies that a collection of samples that are a more realistic representation of the actual input values would produce a more accurate transformation and, therefore, produce a better pdf of the output random variable.

To produce a nonuniform sample based on the $N(0,1)$ distribution of the input random variable, we can generate a set of evenly spaced nodes on $[-a,a]$, (denoted as N), and then evaluate the standard normal CDF, shown in Figure (46), at these nodes (denoted as C). Then the pairs, $\{(c_i,n_i) : c_i \,\epsilon\, C, n_i \,\epsilon\, N, for\ i = 1,\ldots,m\}$, are interpolated, producing a graph of the interpolated inverse CDF. We evaluate the inverse CDF to produce a sample from the Normal[0,1] distribution. To illustrate this method, we generate the inverse CDF, shown in Figure (46), using 500 equally spaced nodes on $[-6,6]$. We then evaluate this inverse CDF at at each node of N where $N = \{1/16, 1/8, 3/16, 1/4, \ldots, 13/16, 7/8, 15/16\}$. The sample generated is shown in Figure (47). In addition to these samples, the endpoints of the interval $[-a,a]$ and two additional points are added to the set in order to capture those areas of the transformation.

The nonuniform sampling method was tested on the nonlinear ODE model problem (24) from Chapter 3. Once the nonuniform sample was generated, the same algorithm to obtain the approximated transformation was used as in the uniform sampling method. Just as the uniform sampling case, when the number of samples increase the accuracy of $\hat{g}(X)$ increases. The graphs in Figure (48) show the increase in accuracy between samples of 35 and 67. The weighted error for 35 and 67 samples are 0.040645

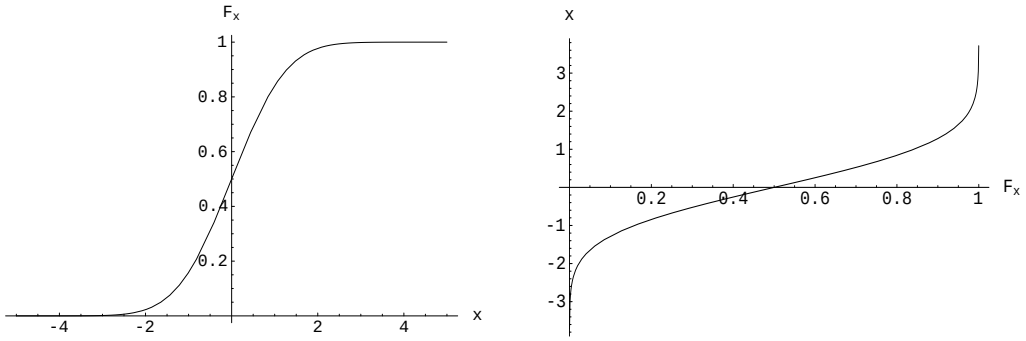


Figure 46: CDF and Inverse CDF of N(0,1) Distribution.

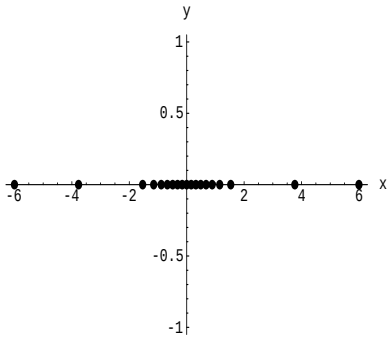


Figure 47: Nonuniform Sample Generated Based on N(0,1) Distribution.

and 0.0230532 respectively. The graph in Figure (49) shows the weighted error for each sample set analyzed. The Mathematica code used for the nonuniform sampling method can be found in Appendix B.3.

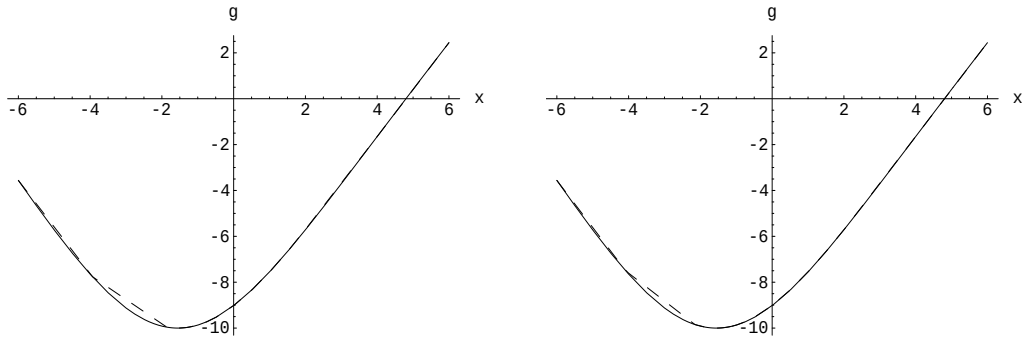


Figure 48: Approximate (dashed) and Actual (solid) Transformation of $Y = g(X)$ Using Nonuniform Sampling Method With 35 and 67 Samples for Nonlinear ODE.

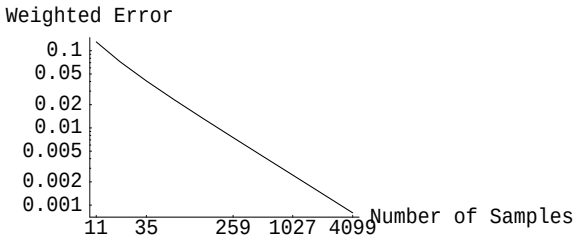


Figure 49: Log-Log Weighted Error Plot by Number of Nonuniform Samples for Nonlinear ODE.

In Table (10), the accuracy of the uniform and nonuniform sampling methods are compared for this nonlinear example. We see that the uniform sampling method produces more accurate results at each level of samples. This loss of accuracy can be attributed to the lack of points near the ends of the interval $[-a, a]$ and where the second derivative of the transformation is non zero. These improvements could greatly reduce the error, but will not be covered here. Further research needs to be conducted to explore and improve this idea of nonuniform sampling.

Nonuniform Sampling		**Uniform Sampling**	
Samples	**Error**	**Samples**	**Error**
35	0.040645	32	0.00802643
67	0.0230532	64	0.00200724
131	0.0131352	128	0.000501754
259	0.0075023	256	0.000125043
515	0.0042869	512	0.0000275471

Table 10: **Comparison of Weighted Error Between Nonuniform and Uniform Sampling Methods Used to Approximate the Transformation of the Nonlinear ODE**

VI. Conclusion and Further Research

Uncertainty analysis plays an important role in engineering design and analysis. For a long time, Monte Carlo simulations and stochastic projection methods have been used to determine output distributions of these systems. These methods, while effective, can be cumbersome and inefficient.

The introduction of sampling combined with the transformation method can open up new doors in determining crucial information about random systems. As shown in this research, the uniform sampling method was an effective way to approximate the transformation function of the linear and nonlinear ordinary differential equation model problems. This method is also quite simple to execute. In chapter 3, the methodology for the Hermite-chaos was shown for the linear and nonlinear ODEs. The calculations required were complicated and consumed a great amount of time to complete. The uniform sampling method, however, was simple and easy to calculate without sacrificing accuracy. Monte Carlo simulation was also simple to conduct, but the limited accuracy and the necessity of running large amounts of random samples through the system make it less appealing. By using the transformation method, finding the distribution was faster and more accurate than Monte Carlo Simulation.

While the uniform sampling and transformation method outperformed Hermite-chaos and Monte Carlo simulation for the two specific examples discussed in chapter 3, more research needs to be conducted. This idea can be extended to systems with more than one random input. The uniform sampling method should also be tested on real world applications. The notion of nonuniform sampling can also be extended and improved to validate its effectiveness. In addition, finite elements could be used to approximate the transformation and the results could be compared to sampling results.

The sampling and transformation method, evaluated in this research, can serve as a benchmark for future improvements in the field of uncertainty analysis. As mentioned before, many advancements can be made on this very simple idea. System reliability, interaction analysis, characteristic determination, and many other important applications can be simplified if the sampling and transformation method idea was further proven to be an effective analytic tool.

Appendix A. ***Mathematica Code for the Linear ODE to Approximate the Transformation***

A.1 Hermite-chaos of Degree 4

```
<<Statistics`ContinuousDistributions`
<<Graphics`Graphics`
npdf[t_] = PDF[NormalDistribution[0, 1], t];
f[t_] = npdf[t];
```

Define the parameters and conditions of the linear ODE.

```
a = −6;
b = 6;
pd = 4;
m = 10;
kb = 10;
kt = 2;
c = 4;
v = 25;
p[t_, i_]:=HermiteH[i, t/√2]/√(2^i i!);
```

Create A and B matrices for S.

```
ipa = IdentityMatrix[pd + 1];
MatrixForm[ipa];
ipb = Table[0, {i, 0, pd}, {j, 0, pd}];
For[i = 0, i ≤ pd, i++,
For[j = i, j ≤ pd, j++,
ipb[[i + 1, j + 1]] = Chop[NIntegrate[tp[t, i]p[t, j]f[t], {t, −Infinity, Infinity}]];
ipb[[j + 1, i + 1]] = ipb[[i + 1, j + 1]];
]
```

```
]
MatrixForm[ipb];
md = Dimensions[ipa][[1]];
aa11 = Table[0, {i, 1, md}, {j, 1, md}];
aa12 = IdentityMatrix[md];
aa21 = -((kb/m)IdentityMatrix[md] + (kt/m)Inverse[ipa].ipb);
aa22 = -(c/m)IdentityMatrix[md];
aa = Chop[Join[Transpose[Join[aa11, aa12]], Transpose[Join[aa21, aa22]]]];
MatrixForm[aa];
```

Solve the system of linear ODEs.

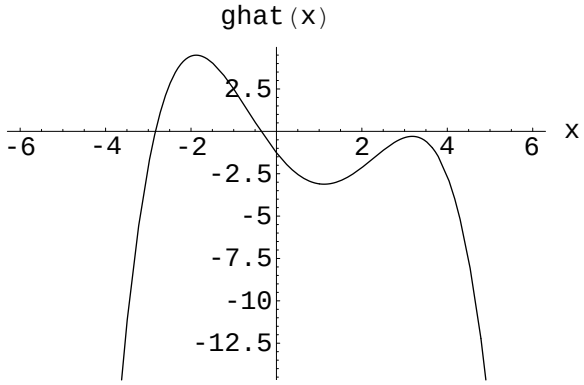
```
tf = 10;
ua = Table[u[i][t], {i, 1, 2md}];
upa = Table[u[i]'[t], {i, 1, 2md}];
ics = Flatten[{Table[u[i][0] == 0, {i, 1, md}], u[md + 1][0] == v,
Table[u[i][0] == 0, {i, md + 2, 2md}]}];
eqns = Flatten[Table[upa[[i]] == (aa.ua)[[i]], {i, 1, 2md}]];
eqns = Flatten[{eqns, ics}];
MatrixForm[eqns];
soln = NDSolve[eqns, Table[u[i], {i, 1, 2md}], {t, 0, tf}];
ut = Table[Evaluate[Table[u[i][t], {i, 1, md}]/.soln], {t, 0, tf}];
Table[Evaluate[Table[u[i][t], {i, md + 1, 2md}]/.soln], {t, 0, tf}];
cc = Flatten[ut[[Dimensions[ut][[1]]]]]
```

$\{-0.537323, -2.02877, 0.514555, 0.941688, -0.584251\}$

Calculate the approximated transformation.

```
as[t_] = Sum[cc[[i]]p[t, i - 1], {i, 1, md}];
TimeUsed[]
Plot[as[t], {t, -6, 6}, AxesLabel → {"x", "ghat(x)"}]
Display["ghat1.eps", %, "EPS"]
```

```
nd = 2^8 + 1;
id = Table[a + (i - 1)(b-a)/(nd-1), {i, 1, nd}];
od = Table[0, {i, 1, nd}];
For[i = 1, i ≤ nd, i++,
od[[i]] = as[id[[i]]]
]
```

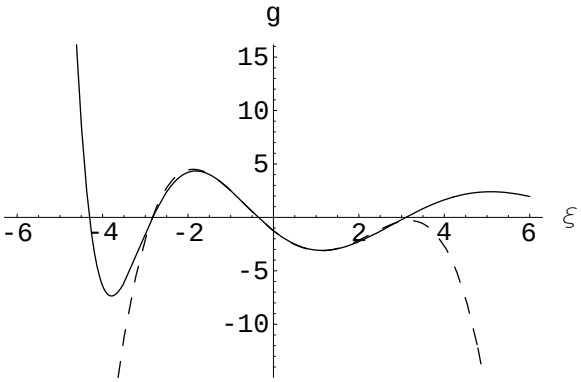


This is the actual transformation for the model problem.

```
qq[s_, t_] = 2v/Sqrt[(c/m)^2 - 4(1/m)(kb + skt)] Exp[-tc/(2m)] Sinh[t/2 Sqrt[(c/m)^2 - 4(1/m)(kb + skt)]];
q[s_] = 2v/Sqrt[(c/m)^2 - 4(1/m)(kb + skt)] Exp[-10c/(2m)] Sinh[10/2 Sqrt[(c/m)^2 - 4(1/m)(kb + skt)]];

Plot[q[s], {s, -6, 6}, AxesLabel → {"x", "g(x)"}]
Display["g.eps", %, "EPS"]
```

Plot the actual and approximate transformations and calculate the weighted error.

```
Plot[{q[t], as[t]}, {t, -6, 6}, AxesLabel → {"ξ", "g"}, PlotStyle → {GrayLevel[0], Dashing[{.03}]}]
Export["gandghat4.eps", %, "EPS"]
WeightedError = Sqrt[NIntegrate[(q[t] - as[t])^2 f[t], {t, -6, 6}]]
```



0.306322

A.2 Uniform Sampling

```
<<Statistics‘ContinuousDistributions‘
<<Graphics‘Graphics‘
```

Define the parameters and conditions of the linear ODE.

```
npdf[t_] = PDF[NormalDistribution[0, 1], t];
f[t_] = npdf[t]
a = −6;
b = 6;
m = 10;
kb = 10;
kt = 2;
c = 4;
v = 25;
tf = 10;
```

This is the actual transformation for the model problem.

$$\mathbf{qq[s\_, t\_] = \frac{2v}{Sqrt[(\frac{c}{m})^2 - 4(\frac{1}{m})(kb + skt)]} Exp[-\frac{tc}{2m}] Sinh[\frac{t}{2} Sqrt[(\frac{c}{m})^2 - 4(\frac{1}{m})(kb + skt)]]}$$

$$\mathbf{q[s\_] = \frac{2v}{Sqrt[(\frac{c}{m})^2 - 4(\frac{1}{m})(kb + skt)]} Exp[-\frac{10c}{2m}] Sinh[\frac{10}{2} Sqrt[(\frac{c}{m})^2 - 4(\frac{1}{m})(kb + skt)]]}$$

$$\frac{50e^{-t/5}\text{Sinh}[\frac{1}{2}\sqrt{\frac{4}{25} - \frac{2}{5}(10+2s)}t]}{\sqrt{\frac{4}{25} - \frac{2}{5}(10+2s)}}$$

$$\frac{50\text{Sinh}[5\sqrt{\frac{4}{25} - \frac{2}{5}(10+2s)}]}{e^2\sqrt{\frac{4}{25} - \frac{2}{5}(10+2s)}}$$

Sample the function defining the transformation from ξ to x(T,ξ) . Use this set of samples to approximate the transformation.

Using 8 Samples,

$$\mathbf{n = 3;}$$

$$\mathbf{nnd = 2^n + 1;}$$

$$\mathbf{h = \frac{b-a}{nnd-1};}$$

$$\mathbf{nd = Table[a + hk, \{k, 0, nnd - 1\}];}$$

$$\mathbf{odata8 = Table[0, \{i, 1, nnd\}];}$$

$$\mathbf{odata8 = Flatten[Table[y1[x]/.NDSolve[\{y1'[t] == y2[t],}$$

$$\mathbf{y2'[t] == -(\frac{kb+nd[[i]]kt}{m})y1[t] - \frac{c}{m}y2[t],}$$

$$\mathbf{y1[0] == 0, y2[0] == v\}, \{y1, y2\}, \{t, 0, tf\}]/.x \to tf, \{i, 1, nnd\}]];}$$

Interpolate the points generated from the uniform sampling method.

$$\mathbf{T8 = Transpose[\{nd, odata8\}];}$$

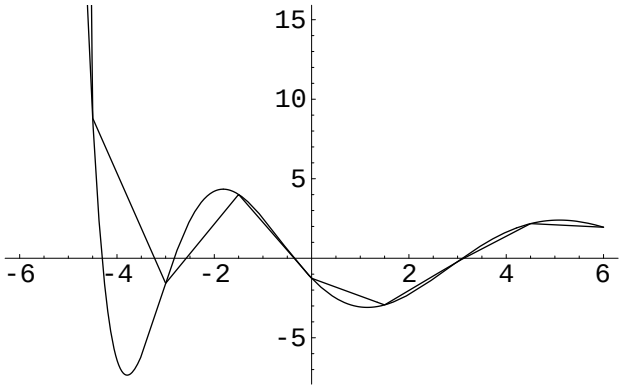
$$\mathbf{pp8 = Interpolation[T8, InterpolationOrder \to 1];}$$

Continue above steps for each level of samples. Plot the actual and approximate transformations and calculate the weighted error for each level of samples.

Using 8 Samples,

$$\mathbf{Plot[\{pp8[t], q[t]\}, \{t, -6, 6\}]}$$

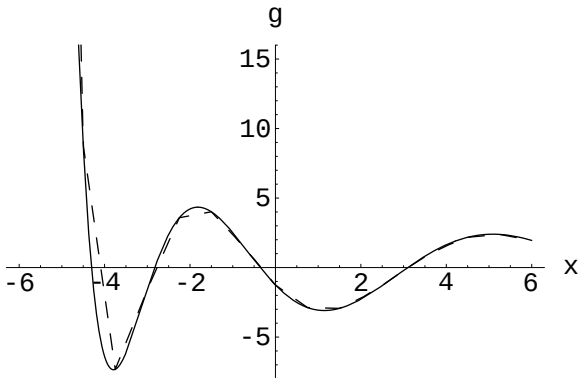
$$\mathbf{WE8 = Sqrt[NIntegrate[(q[t] - pp8[t])^2 f[t], \{t, -6, 6\}]]}$$



0.601852

Using 16 Samples,

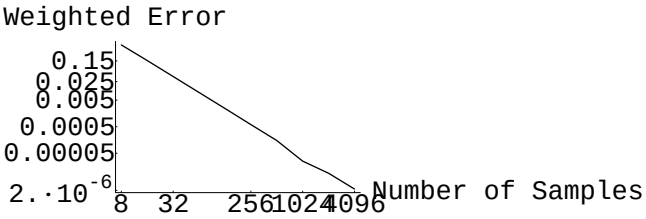
```
Plot[{pp16[t], q[t]}, {t, −6, 6}, AxesLabel → {"x", "g"},
PlotStyle → {Dashing[{.03}], GrayLevel[0]}]
WE16 = Sqrt[NIntegrate[(q[t] − pp16[t])^2 f[t], {t, −6, 6}]]
```



0.155178

Repeat this process for the rest of the sample levels. Produce the error plot of approximated transformation for the linear ODE.

```
LogLogListPlot[{{8, WE8}, {16, WE16}, {32, WE32}, {64, WE64}, {128, WE128}, {256, WE256},
{512, WE512}, {1024, WE1024}, {2048, WE2048}, {4096, WE4096}},
AxesLabel → {"Number of Samples", "Weighted Error"}, PlotJoined → True,
Ticks → {{8, 32, 256, 1024, 4096}, {.005, .025, .15, .0005, .00005, .000002}}]
Export["linsamplingerror.eps", %, "EPS"]
```


Weighted Error
0.15
0.025
0.005
0.0005
0.00005
2.·10^-6
8
32
256
1024
4096
Number of Samples

Appendix B. ***Mathematica Code for the Nonlinear ODE to Approximate the Transformation***

B.1 Hermite-chaos of Degree 4

Define the parameters and conditions of the linear ODE.

```
<<Statistics`ContinuousDistributions`
<<Graphics`Graphics`
a = -Infinity;
b = Infinity;
kb = 1.;
kt = 0.02;
x0 = 10.;
tf = 10;
m = 10.;
k1 = 10.;
k3b = 1;
k3t = 0.02;
v = 0;
npdf[t_] = PDF[NormalDistribution[0, 1], t];
f[t_] = npdf[t]
p[t_, i_]:=HermiteH[i, t/Sqrt[2]]/Sqrt[2^i i!]
E^(-t^2/2)/Sqrt[2 Pi]
```

Calculate the inner products.

```
md = 5;
ip1 = Chop[Table[N[Integrate[p[t, i - 1] p[t, j - 1] p[t, k - 1] p[t, l - 1] f[t], {t, a, b}], 30], {i, 1, md},
{j, 1, md}, {k, 1, md}, {l, 1, md}]];
ip2 = Chop[Table[N[Integrate[tp[t, i - 1] p[t, j - 1] p[t, k - 1] p[t, l - 1] f[t], {t, a, b}], 30], {i, 1, md},
{j, 1, md}, {k, 1, md}, {l, 1, md}]];
```

Solve the system of nonlinear ODEs.

```
deqns = Chop[Table[u[s]''[t] + kl/m u[s][t]+
kb/m Sum[u[i][t]u[j][t]u[k][t]ip1[[i, j, k, s]], {i, 1, md}, {j, 1, md}, {k, 1, md}]+
kt/m Sum[u[i][t]u[j][t]u[k][t]ip2[[i, j, k, s]], {i, 1, md}, {j, 1, md}, {k, 1, md}] == 0, {s, 1, md}]];
ics = Flatten[{u[1][0] == x0, Table[u[i][0] == 0., {i, 2, md}], Table[u[i]'[0] == 0., {i, 1, md}]}];
eqns = Flatten[{deqns, ics}];
MatrixForm[eqns];
soln = NDSolve[eqns, Table[u[i], {i, 1, md}], {t, 0, tf}, Method → Adams];
ut = Table[Evaluate[Table[u[i][t], {i, 1, md}]/.soln], {t, 0, tf}];
cc = Flatten[ut[[Dimensions[ut][[1]]]]];
```

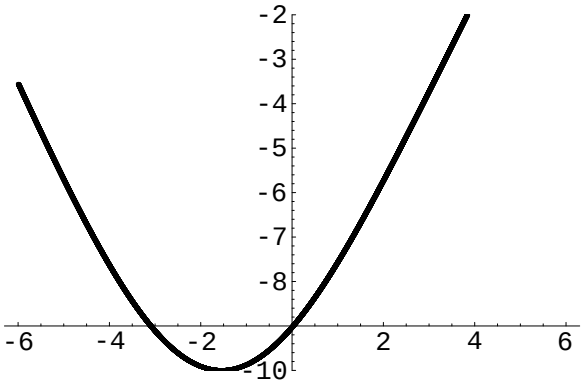
Calculate the approximated transformation.

```
as[t_] = Sum[cc[[i]]p[t, i − 1], {i, 1, md}]
```

$-8.7142+1.09115t+0.146231(-2+2t^2)-0.0114968(-6\sqrt{2}t+2\sqrt{2}t^3)-0.00124268(12-24t^2+4t^4)$

Calculate the actual transformation with 8192 samples.

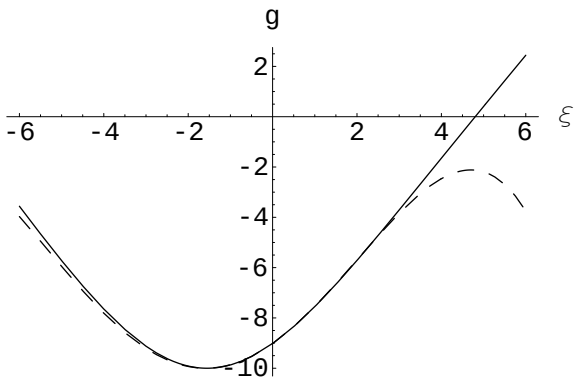
```
a = −6;
b = 6;
n = 13;
nnd = 2^13 + 1;
h = (b−a)/(nnd−1);
nd = Table[a + hk, {k, 0, nnd − 1}];
odata = Table[0, {i, 1, nnd}];
odata = Flatten[Table[y1[x]/.NDSolve[{y1'[t] == y2[t],
y2'[t] == −((k3b+nd[[i]]k3t)/m)((y1[t])^3) − kl/m y1[t],
y1[0] == 10, y2[0] == v}, {y1, y2}, {t, 0, tf}]/.x → tf, {i, 1, nnd}]];
ListPlot[Transpose[{nd, odata}], PlotRange → {−2, −10}]
Display["nonlinsamptransactual.eps", %, "EPS"]
```




```
T = Transpose[{nd, odata}];
pp = Interpolation[T, InterpolationOrder → 1];
```

Plot the actual and approximate transformations and calculate the weighted error.

```
Plot[{as[t], pp[t]}, {t, −6, 6}, AxesLabel → {"ξ", "g"},
PlotStyle → {Dashing[{.03}], GrayLevel[0]}]
Export["gandghatnl4.eps", %, "EPS"]
WeightedError = Sqrt[NIntegrate[(pp[t] − as[t])^2 f[t], {t, −6, 6}]]
```



0.0164006

B.2 Uniform Sampling

Define the parameters and conditions of the nonlinear ODE.

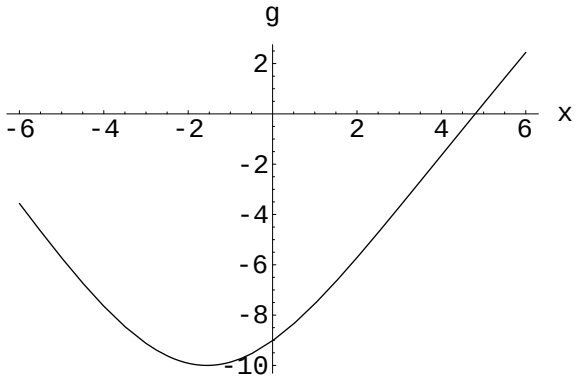
```
<<Statistics‘ContinuousDistributions‘
<<Graphics‘Graphics‘
```

```
a = -6;
b = 6;
kb = 1.;
kt = 0.02;
tf = 10;
m = 10.;
k1 = 10.;
k3b = 1;
k3t = 0.02;
v = 0;
npdf[t_] = PDF[NormalDistribution[0, 1], t];
f[t_] = npdf[t]
```

$$\frac{e^{-\frac{t^2}{2}}}{\sqrt{2\pi}}$$

Calculate the actual transformation with 8192 samples.

```
n = 13;
nnd = 2^n + 1;
h = (b - a)/(nnd - 1);
nd = Table[a + h k, {k, 0, nnd - 1}];
odata = Table[0, {i, 1, nnd}];
odata = Flatten[Table[y1[x]/.NDSolve[{y1'[t] == y2[t],
y2'[t] == -((k3b + nd[[i]] k3t)/m)((y1[t])^3) - (k1/m) y1[t],
y1[0] == 10, y2[0] == v}, {y1, y2}, {t, 0, tf}]/.x -> tf, {i, 1, nnd}]];
T = Transpose[{nd, odata}];
pp = Interpolation[T, InterpolationOrder -> 1];
Plot[pp[t], {t, -6, 6}, AxesLabel -> {"x", "g"}]
Export["nlactualtrans.eps", %, "EPS"]
```



Sample the function defining the transformation from ξ to x(T,ξ) . Use this set of samples to approximate the transformation.

Using 8 Samples,

$n = 3;$

$\text{nnd} = 2^n + 1;$

$h = \frac{b-a}{\text{nnd}-1};$

$\text{nd} = \text{Table}[a + hk, \{k, 0, \text{nnd} - 1\}];$

$\text{odata8} = \text{Table}[0, \{i, 1, \text{nnd}\}];$

$\text{odata8} = \text{Flatten}[\text{Table}[\text{y1}[x]/.\text{NDSolve}[\{\text{y1}'[t] == \text{y2}[t],$
$\text{y2}'[t] == -(\frac{\text{k3b+nd[[i]]k3t}}{m})((\text{y1}[t])\text{^}3) - \frac{\text{k1}}{m}\text{y1}[t],$
$\text{y1}[0] == 10, \text{y2}[0] == v\}, \{\text{y1}, \text{y2}\}, \{t, 0, \text{tf}\}]/.x \to \text{tf}, \{i, 1, \text{nnd}\}]];$

Interpolate the points generated from the uniform sampling method.

$\text{T8} = \text{Transpose}[\{\text{nd}, \text{odata8}\}];$

$\text{pp8} = \text{Interpolation}[\text{T8}, \text{InterpolationOrder} \to 1];$

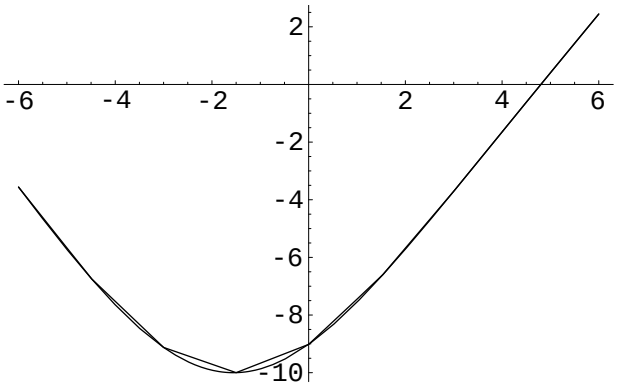
$\text{Plot}[\text{pp8}[t], \{t, -6, 6\}]$

Continue above steps for each level of samples. Plot the actual and approximate transformations and calculate the weighted error for each level of samples.

Using 8 Samples,

Plot[{pp8[t], pp[t]}, {t, −6, 6}]

WE8 = Sqrt[NIntegrate[(pp[t] − pp8[t])^2 f[t], {t, −6, 6}]]



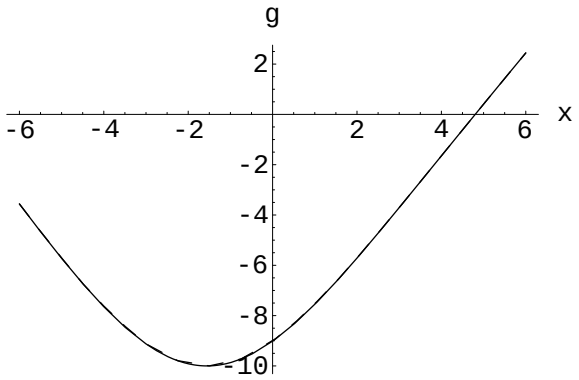
0.127753

Using 16 Samples,

Plot[{pp16[t], pp[t]}, {t, −6, 6}, AxesLabel → {"x", "g"},

PlotStyle → {Dashing[{.03}], GrayLevel[0]}]

WE16 = Sqrt[NIntegrate[(pp[t] − pp16[t])^2 f[t], {t, −6, 6}]]



0.032066

Repeat this process for the rest of the sample levels. Produce the error plot for the nonlinear ODE.

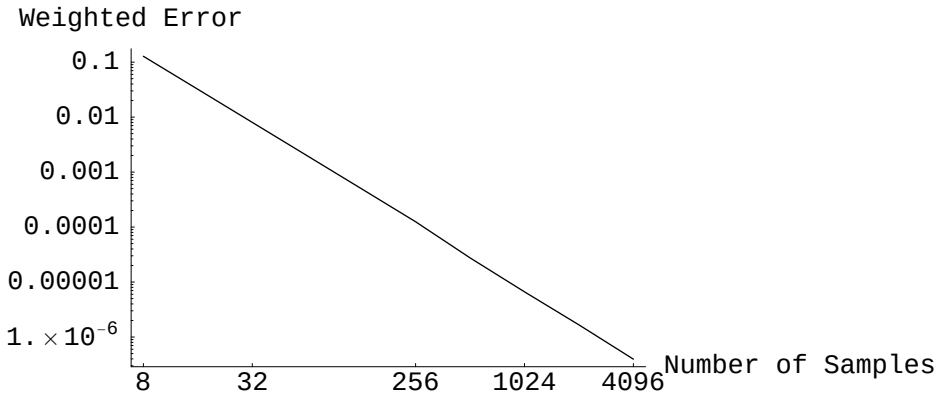
LogLogListPlot[{{8, WE8}, {16, WE16}, {32, WE32}, {64, WE64}, {128, WE128},

, {256, WE256}, {512, WE512}, {1024, WE1024}, {2048, WE2048}, {4096, WE4096}},

AxesLabel → {"Number of Samples", "Weighted Error"}, PlotJoined → True,

Ticks → {{8, 32, 256, 1024, 4096}, Automatic}]

Export["nlsamplingerror.eps", %, "EPS"]



B.3 Nonuniform Sampling

Define the parameters and conditions of the nonlinear ODE.

<<Statistics‘ContinuousDistributions‘

<<Graphics‘Graphics‘

$a = -6;$

$b = 6;$

kb = 1.;

kt = 0.02;

tf = 10;

$m = 10.;$

k1 = 10.;

k3b = 1;

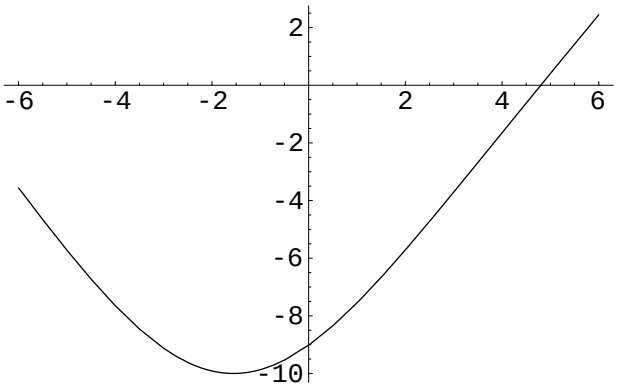
k3t = 0.02;

$v = 0;$

```
dist = NormalDistribution[0, 1];
pdf[x_] = PDF[dist, x];
cdf[x_] = CDF[dist, x];
```

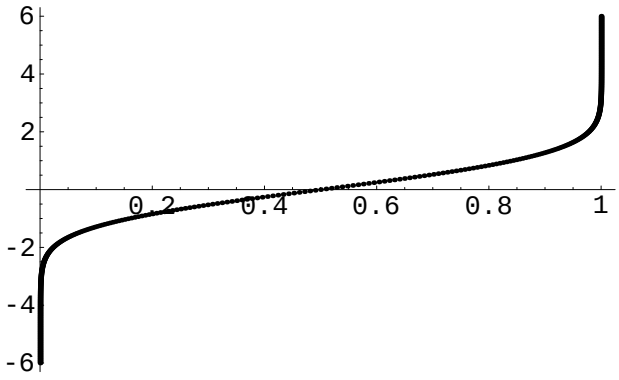
Calculate the actual transformation with 8192 samples.

```
n = 13;
nnd = 2^n + 1;
h = (b - a)/(nnd - 1);
id = Table[a + hk, {k, 0, nnd - 1}];
od = Table[0, {i, 1, nnd}];
od = Flatten[Table[y1[x]/.NDSolve[{y1'[t] == y2[t],
y2'[t] == -((k3b + id[[i]] k3t)/m)((y1[t])^3) - (k1/m) y1[t],
y1[0] == 10, y2[0] == v}, {y1, y2}, {t, 0, tf}]/.x → tf,
{i, 1, nnd}]];
T = Transpose[{id, od}];
pp = Interpolation[T, InterpolationOrder → 1];
Plot[pp[t], {t, -6, 6}]
```



Calculate the inverse cdf of the normal distribution.

```
in = 500;
s = Table[N[{cdf[-6 + 12i/in], -6 + 12i/in}], {i, 0, in}];
ListPlot[s]
cdfi = Interpolation[s, InterpolationOrder → 1];
```



Evaluate the inverse cdf to get a nonuniform sample from the normal distribution.

Using 11 Samples,

$k = 3;$

$n = 2^k - 1;$

$\text{xx} = \text{Table}[\frac{i}{n+1}, \{i, 1, n\}];$

$y = \text{Table}[0, \{i, 1, n\}];$

$y = \text{cdfi}[\text{xx}];$

$\text{nd} = \text{Table}[0, \{i, 1, n\}];$

$\text{nd} = \text{Flatten}[\{a, \frac{(y[[1]]-6)}{2}, y, \frac{y[[\text{Length}[y]]]+6}{2}, b\}]$

$\{-6, -3.57519, -1.15037, -0.674508,$

$-0.318658, 0., 0.318658, 0.674508, 1.15037, 3.57519, 6\}$

Sample the function defining the transformation from ξ to x(T,ξ) . Use this set of samples to approximate the transformation.

$\text{nnd} = \text{Length}[\text{nd}]$

$\text{odata} = \text{Flatten}[\text{Table}[\text{y1}[z]/.\text{NDSolve}[\{\text{y1}'[x] == \text{y2}[x],$

$\text{y2}'[x] == -(\frac{\text{k3b+nd[[i]]k3t}}{m})((\text{y1}[x])\text{\^{}}3) - \frac{\text{k1}}{m}\text{y1}[x],$

$\text{y1}[0] == 10, \text{y2}[0] == v\}, \{\text{y1}, \text{y2}\}, \{x, 0, \text{tf}\}]/.z \to \text{tf}, \{i, 1, \text{nnd}\}]];$

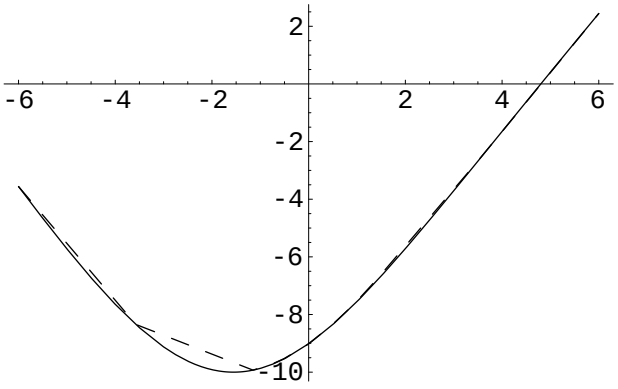
Interpolate the points generated from the nonuniform sampling method.

```
T11 = Transpose[{nd, odata}];
pp11 = Interpolation[T11, InterpolationOrder → 1];
Plot[pp11[t], {t, −6, 6}];
```

Continue above steps for each level of samples. Plot the actual and approximate transformations and calculate the weighted error for each level of samples.

Using 11 Samples,

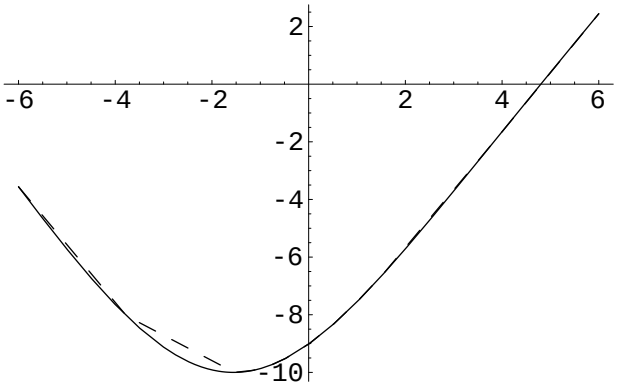
```
Plot[{pp11[t], pp[t]}, {t, −6, 6},
PlotStyle → {Dashing[{.03}], GrayLevel[0]}]
WE11 = Sqrt[NIntegrate[(pp[t] − pp11[t])^2 pdf[t], {t, −6, 6}]]
```



0.130374

Using 19 Samples,

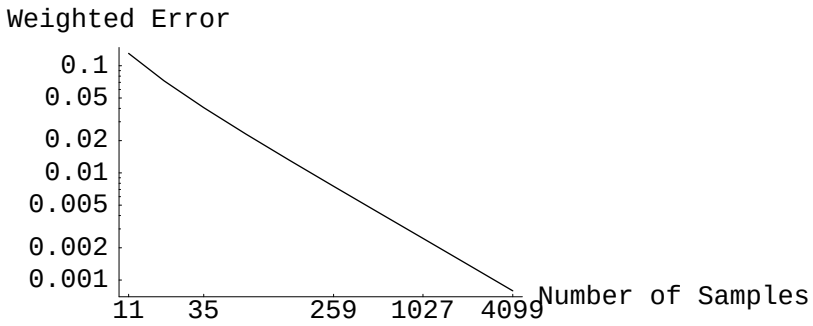
```
Plot[{pp19[t], pp[t]}, {t, −6, 6},
PlotStyle → {Dashing[{.03}], GrayLevel[0]}]
WE19 = Sqrt[NIntegrate[(pp[t] − pp19[t])^2 pdf[t], {t, −6, 6}]]
```



0.0722684

Repeat this process for the rest of the sample levels. Produce the error plot for the nonlinear ODE.

```
LogLogListPlot[{{11, WE11}, {19, WE19}, {35, WE35}, {67, WE67},
{131, WE131}, {259, WE259}, {515, WE515}, {1027, WE1027},
{2051, WE2051}, {4099, WE4099}},
AxesLabel → {"Number of Samples", "Weighted Error"}, PlotJoined → True,
Ticks → {{11, 35, 259, 1027, 4099}, Automatic}]
Export["nlnonuniformsamplingerror.eps", %, "EPS"]
```



Appendix C. ***Mathematica Code for the CDF Transformation Method***

Define the parameters and conditions of the linear ODE.

```
<<Statistics`ContinuousDistributions`
<<Graphics`Graphics`
npdf[t_] = PDF[NormalDistribution[0, 1], t];
f[t_] = npdf[t];
a = -6;
b = 6;
pd = 4;
tf = 10;
m = 10;
kb = 10;
kt = 2;
c = 4;
v = 25;
p[t_, i_]:=HermiteH[i, t/Sqrt[2]]/Sqrt[2^i i!];
```

Create A and B matrices for S.

```
ipa = IdentityMatrix[pd + 1];
MatrixForm[ipa];
ipb = Table[0, {i, 0, pd}, {j, 0, pd}];
For[i = 0, i <= pd, i++,
For[j = i, j <= pd, j++,
ipb[[i + 1, j + 1]] = Chop[NIntegrate[tp[t, i]p[t, j]f[t], {t, -Infinity, Infinity}]];
ipb[[j + 1, i + 1]] = ipb[[i + 1, j + 1]];
]
]
```

```
MatrixForm[ipb];
md = Dimensions[ipa][[1]];
aa11 = Table[0, {i, 1, md}, {j, 1, md}];
aa12 = IdentityMatrix[md];
aa21 = -((kb/m)IdentityMatrix[md] + (kt/m)Inverse[ipa].ipb);
aa22 = -(c/m)IdentityMatrix[md];
aa = Chop[Join[Transpose[Join[aa11, aa12]], Transpose[Join[aa21, aa22]]]];
MatrixForm[aa];
```

Solve the system of linear ODEs.

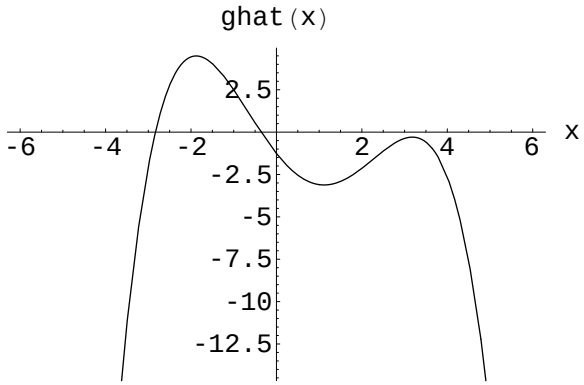
```
ua = Table[u[i][t], {i, 1, 2md}];
upa = Table[u[i]'[t], {i, 1, 2md}];
ics = Flatten[{Table[u[i][0] == 0, {i, 1, md}], u[md + 1][0] == v,
Table[u[i][0] == 0, {i, md + 2, 2md}]}];
eqns = Flatten[Table[upa[[i]] == (aa.ua)[[i]], {i, 1, 2md}]];
eqns = Flatten[{eqns, ics}];
MatrixForm[eqns];
soln = NDSolve[eqns, Table[u[i], {i, 1, 2md}], {t, 0, tf}];
ut = Table[Evaluate[Table[u[i][t], {i, 1, md}]/.soln], {t, 0, tf}];
Table[Evaluate[Table[u[i][t], {i, md + 1, 2md}]/.soln], {t, 0, tf}];
cc = Flatten[ut[[Dimensions[ut][[1]]]]]
```

$\{-0.537323, -2.02877, 0.514555, 0.941688, -0.584251\}$

Calculate the approximated transformation.

```
as[t_] = Sum[cc[[i]]p[t, i - 1], {i, 1, md}];
Plot[as[t], {t, -6, 6}, AxesLabel → {"x", "ghat(x)"}]
nd = 2^12 + 1;
id = Table[a + (i - 1)(b-a)/(nd-1), {i, 1, nd}];
od = Table[0, {i, 1, nd}];
For[i = 1, i ≤ nd, i++,
```

```
od[[i]] = as[id[[i]]]
]
```



Using approximate transformation just found, generate the pdf using the transformation method.

```
exlim = Table[{0, 0, 0, 0, 0}, {i, 1, nd}];
cnt = 1;
exlim[[cnt]] = {1, id[[cnt]], od[[cnt]], 1, 0};
index = 2;
If[od[[index]] − od[[index − 1]] < 0, dir = −1, dir = 1];
While[index < nd,
index++;
δ = Sign[od[[index]] − od[[index − 1]]];
If[δ == dir, ,
cnt++;
exlim[[cnt]] = {index − 1, id[[index − 1]], od[[index − 1]], 1, 0};
dir = −Sign[dir]
];
]
cnt++;
exlim[[cnt]] = {index, id[[index]], od[[index]], 1, 0};
```

```
exlim = Take[exlim, cnt];
If[exlim[[1, 3]] > exlim[[2, 3]],
exlim[[1, 5]] = 1; exlim[[2, 5]] = -1,
exlim[[2, 5]] = 1; exlim[[1, 5]] = -1
];
For[i = 3, i ≤ cnt, i++,
exlim[[i, 5]] = If[exlim[[i - 1, 5]] == 1, -1, 1]
];
nex = Dimensions[exlim][[1]];
If[od[[index]] - od[[index - 1]] < 0,
max = Take[exlim, {1, cnt, 2}];
min = Take[exlim, {2, cnt - 1, 2}],
min = Take[exlim, {1, cnt, 2}];
max = Take[exlim, {2, cnt - 1, 2}]
];
max = Sort[max, #1[[2]] < #2[[2]]&];
min = Sort[min, #1[[2]] < #2[[2]]&];
nmax = Dimensions[max][[1]];
nmin = Dimensions[min][[1]];
node = N[Transpose[{Table[i, {i, 1, nd}], id, od, Table[0, {i, 1, nd}]}]];
ysnode = Sort[node, #1[[3]] < #2[[3]]&];
exlim = Chop[exlim]
(*remove repeated yvalues from ysnode*)
new = {ysnode[[1]]};
For[i = 1, i < Dimensions[ysnode][[1]], i++,
If[Chop[ysnode[[i, 3]]==ysnode[[i + 1, 3]]],
(*do nothing*),
(*add to new*)new = Join[new, {ysnode[[i + 1]]}]
```

]

]

$\{\{1,-6,-180.908,1,-1\},\{1408,-\frac{1923}{1024},4.49415,1,1\},$
$\{2430,\frac{1143}{1024},-3.11646,1,-1\},\{3135,\frac{1629}{512},-0.295545,1,1\},\{4097,6,-53.0141,1,-1\}\}$

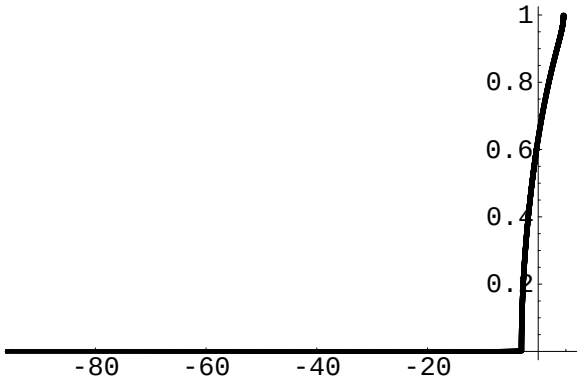
For [(* start at $i = 2$ because $i = 1$ is the minimum in new so there is no contribution ,
then process each element of new *)
$i = 2, i \leq$ Dimensions[new][[1]], i++,
$j = 1$;
(*nex is the number of entries in exlim , which contains the local extrema
and/or the limits of integration *)
While [$j \leq$ nex,
(* Print["j = ", j, " nex = ", nex]; *)
If[(* an extreme *)exlim[[j, 4]] == 1,
If[(*a minimum*)exlim[[j, 5]] == -1,
(*Print["i = ", i, " j = ", j, " min"]; *)
If[new[[i, 3]] > exlim[[j, 3]],
(*add limits and remove min *)
tm = exlim[[j]];
If[tm[[1]] $\neq$ 1,
xtmp = (node[[tm[[1]] $-$ 1, 2]] $-$ node[[tm[[1]], 2]])/(node[[tm[[1]] $-$ 1, 3]] $-$ node[[tm[[1]], 3]])
(new[[i, 3]] $-$ node[[tm[[1]], 3]]) + node[[tm[[1]], 2]];
tmp = {{tm[[1]], xtmp, new[[i, 3]], 0, -1}};
exlim = Flatten[{Take[exlim, $j-1$], tmp, Take[exlim, $j-$ nex]}, 1],
tmp = {{tm[[1]], node[[tm[[1]], 2]], new[[i, 3]], 0, -1}};
exlim = Flatten[{Take[exlim, $j-1$], tmp, Take[exlim, $j-$ nex]}, 1]
];
(*Print[exlim]; *)
If[tm[[1]] $\neq$ nd,

```
xtmp = (node[[tm[[1]] + 1, 2]] - node[[tm[[1]], 2]])/(node[[tm[[1]] + 1, 3]] - node[[tm[[1]], 3]])
(new[[i, 3]] - node[[tm[[1]], 3]]) + node[[tm[[1]], 2]];
tmp = {{tm[[1]], xtmp, new[[i, 3]], 0, 1}};
exlim = Flatten[{Take[exlim, j], tmp, Take[exlim, j - nex]}, 1],
tmp = {{tm[[1]], node[[tm[[1]], 2]], new[[i, 3]], 0, 1}};
exlim = Flatten[{Take[exlim, j], tmp, Take[exlim, j - nex]}, 1]
];
(*Print[exlim]; *)
nex++
,
(*do nothing*)
]
, (*else a maximum*)
(*If[(i > 192)&&(i < 195), Print["i = ", i, " j = ", j, " max", " nex = ", nex];
Print[exlim];
Print[new[[i, 3]], " ", exlim[[j, 3]]],
]; *)
If[new[[i, 3]] ≥ exlim[[j, 3]],
(*removelimitsand max *)
If[(nex > 3)&&(j ≠ 1)&&(j ≠ nex),
exlim = Flatten[{Take[exlim, j - 2], Take[exlim, j + 1 - nex]}, 1];
nex = nex - 3;
j = j - 3,
If[(j ≠ 1)&&(j ≠ nex),
exlim = Take[exlim, -2];
nex = 2,
If[j == 1,
exlim = Flatten[{Take[exlim, 1], Take[exlim, 2 - nex]}, 1];
nex = nex - 1;
```

```
exlim[[1, 4]] = 0;
exlim[[1, 5]] = −1,
If[j == nex,
exlim = Flatten[{Take[exlim, nex − 2], Take[exlim, −1]}, 1];
nex = nex − 1;
exlim[[nex, 4]] = 0;
exlim[[nex, 5]] = 1,
](*endIf[j == nex...*)
](*endIf[j == 1...*)
](*If[j ≠ 1...*)
](*endIf[nex > 3...*)
,
(*else do nothing*)
](*end If[new[[i, 3]]...*)
](*end If (a minimum )*)
]; (*end If (an extreme )*)
j++
]; (*end While[j ≤ nex*)
(*Interpolate to define limits of integration*)
For[j = 1, j ≤ nex, j++,
If[(*a limit*)exlim[[j, 4]] == 0,
If[(*left limit*)exlim[[j, 5]] == −1,
While[((exlim[[j, 1]] > 1)&&(new[[i, 3]] ≥ node[[exlim[[j, 1]] − 1, 3]])),
exlim[[j, 1]] = exlim[[j, 1]] − 1
](*end While[((exlim...*);
in = exlim[[j, 1]];
If[in ≠ 1,
exlim[[j, 2]] = (node[[in − 1, 2]] − node[[in, 2]])/(node[[in − 1, 3]] − node[[in, 3]])
(new[[i, 3]] − node[[in, 3]]) + node[[in, 2]];
```

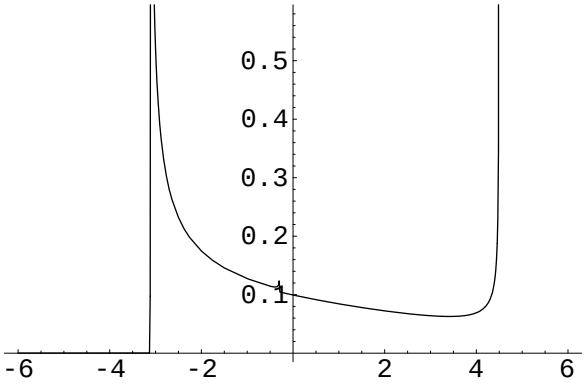
```
exlim[[j,3]] = new[[i,3]],
exlim[[j,2]] = node[[in,2]];
exlim[[j,3]] = new[[i,3]]
](*end If[in ≠ 1*)
,(*else a right limit*)
While[((exlim[[j,1]] < nd)&&(new[[i,3]] ≥ node[[exlim[[j,1]] + 1,3]])),
exlim[[j,1]] = exlim[[j,1]] + 1
];
in = exlim[[j,1]];
If[in ≠ nd,
exlim[[j,2]] = (node[[in + 1,2]] − node[[in,2]])/(node[[in + 1,3]] − node[[in,3]])
(new[[i,3]] − node[[in,3]]) + node[[in,2]];
exlim[[j,3]] = new[[i,3]],
exlim[[j,2]] = node[[in,2]];
exlim[[j,3]] = new[[i,3]]
]
](*end If[(*left limit*)...*)
,
(*do nothing*)
](*end If[(*a limit*)...*)
];(*end For[j = 1...*)
(*integrate and update cdf*)
j = 1;
While[j ≤ nex,
While[((exlim[[j,4]] ≠ 0)&&(j < nex)),
j++
];
(*If[i > 220,Print["integrate ","i = ",i," j = ",j," ",new[[i,4]]," ",exlim],];*)
If[exlim[[j,4]] == 0,
```

```
new[[i, 4]] = new[[i, 4]] + NIntegrate[npdf[x], {x, exlim[[j, 2]], exlim[[j + 1, 2]]}],
];
j = j + 2
]
];
(*end For i*)


cdf = Transpose[Take[Transpose[new], -2]];
ListPlot[cdf]
```



Differentiate the cdf to get the pdf.

```
pdf = Table[{cdf[[i, 1]], (cdf[[i + 1, 2]] - cdf[[i - 1, 2]])/(cdf[[i + 1, 1]] - cdf[[i - 1, 1]]) },
{i, 2, Dimensions[cdf][[1]] - 1}];


pp = Interpolation[pdf, InterpolationOrder → 1];
Plot[{pp[t]}, {t, -6, 6}]
```



This is the actual transformation for the model problem.

$$\text{qq[s\_, t\_]} = \frac{2v}{\text{Sqrt}[(\frac{c}{m})^2 - 4(\frac{1}{m})(\text{kb} + s\text{kt})]}\text{Exp}[-\frac{tc}{2m}]\text{Sinh}[\frac{t}{2}\text{Sqrt}[(\frac{c}{m})^2 - 4(\frac{1}{m})(\text{kb} + s\text{kt})]];$$

$$q\text{[s\_]} = \frac{2v}{\text{Sqrt}[(\frac{c}{m})^2 - 4(\frac{1}{m})(\text{kb} + s\text{kt})]}\text{Exp}[-\frac{10c}{2m}]\text{Sinh}[\frac{10}{2}\text{Sqrt}[(\frac{c}{m})^2 - 4(\frac{1}{m})(\text{kb} + s\text{kt})]];$$

$$\text{Plot}[\{q[t], \text{as}[t]\}, \{t, -6, 6\}, \text{AxesLabel} \rightarrow \{\text{"x"}, \text{"g"}\}, \text{PlotStyle} \rightarrow \{\text{GrayLevel}[0], \text{Dashing}[\{.03\}]\}]$$

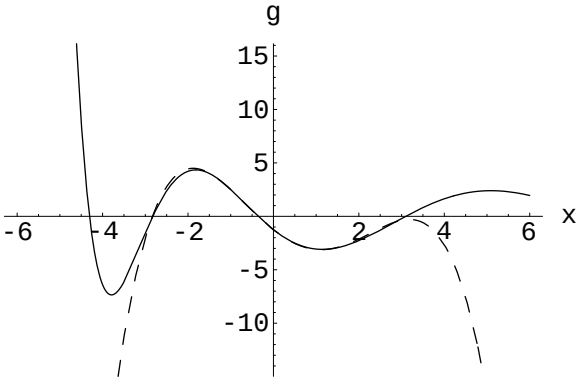
$$\text{id} = \text{Table}[a + (i-1)\frac{b-a}{\text{nd}-1}, \{i, 1, \text{nd}\}];$$

$$\text{od} = \text{Table}[0, \{i, 1, \text{nd}\}];$$

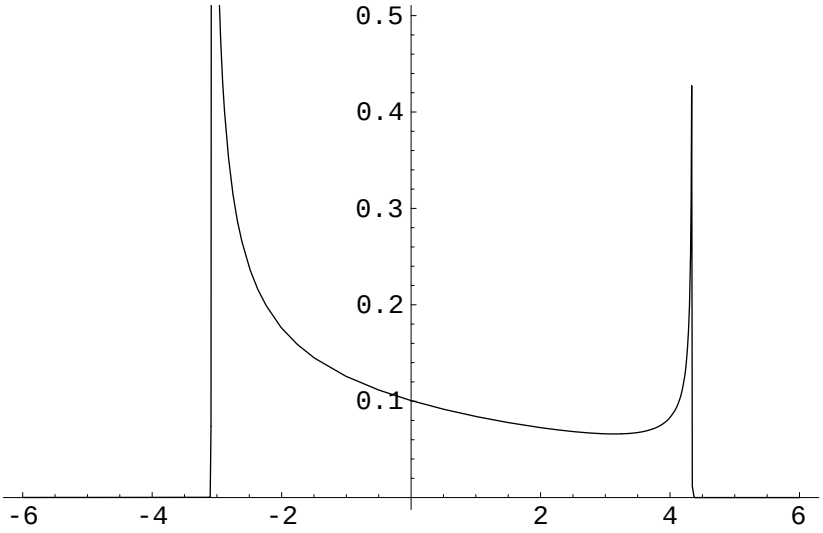
$$\text{For}[i = 1, i \leq \text{nd}, i{+}{+},$$

$$\text{od}[[i]] = q[\text{id}[[i]]]$$

$$]$$

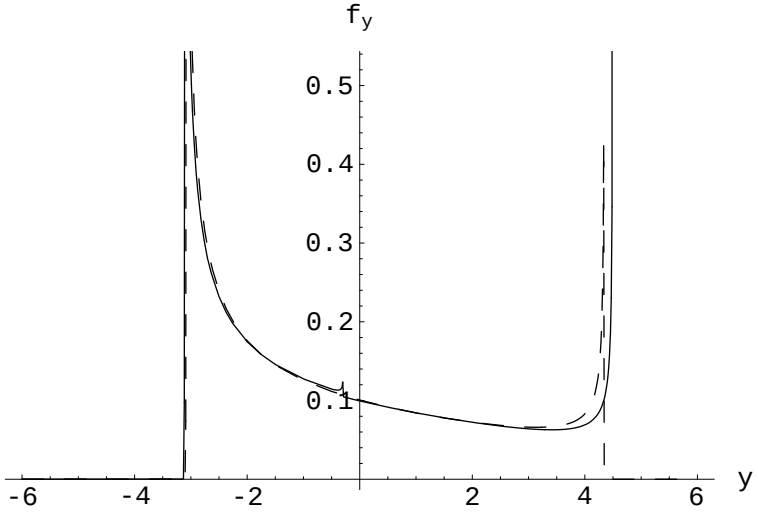


Find the cdf of the actual transformation with the transformation method using the same code used to approximate the transformation.



Plot the actual and approximate pdfs and calculate the error.

```
Plot[{pp[t], pp1[t]}, {t, -6, 6}, AxesLabel → {"y", "f_y"},
PlotStyle → {GrayLevel[0], Dashing[{.03}]}]
Display["lingandghat4pdf.eps", %, "EPS"]
WeightedError = Sqrt[NIntegrate[(pp[t] - pp1[t])^2, {t, -3, 4.1}]]
```



0.0228001

Define the parameters and conditions of the linear ODE.

```
<<Statistics`ContinuousDistributions`
<<Graphics`Graphics`
npdf[t_] = PDF[NormalDistribution[0, 1], t];
f[t_] = npdf[t];
a = -6;
b = 6;
pd = 4;
tf = 10;
m = 10;
kb = 10;
kt = 2;
c = 4;
v = 25;
p[t_, i_]:=HermiteH[i, t/Sqrt[2]]/Sqrt[2^i i!];
```

Create A and B matrices for S.

```
ipa = IdentityMatrix[pd + 1];
MatrixForm[ipa];
ipb = Table[0, {i, 0, pd}, {j, 0, pd}];
For[i = 0, i ≤ pd, i++,
For[j = i, j ≤ pd, j++,
ipb[[i + 1, j + 1]] = Chop[NIntegrate[tp[t, i]p[t, j]f[t], {t, -Infinity, Infinity}]];
ipb[[j + 1, i + 1]] = ipb[[i + 1, j + 1]];
]
]
MatrixForm[ipb];
md = Dimensions[ipa][[1]];
```

```
aa11 = Table[0, {i, 1, md}, {j, 1, md}];
aa12 = IdentityMatrix[md];
aa21 = -((kb/m)IdentityMatrix[md] + (kt/m)Inverse[ipa].ipb);
aa22 = -(c/m)IdentityMatrix[md];
aa = Chop[Join[Transpose[Join[aa11, aa12]], Transpose[Join[aa21, aa22]]]];
MatrixForm[aa];
```

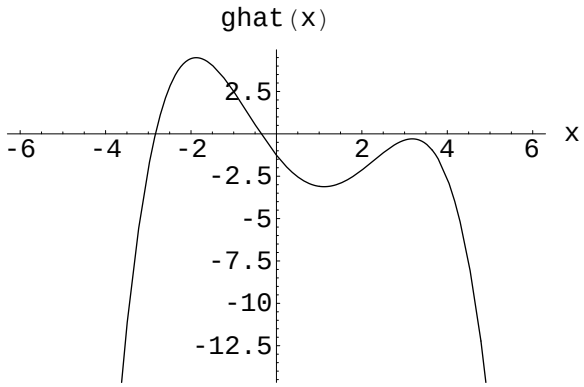
Solve the system of linear ODEs.

```
ua = Table[u[i][t], {i, 1, 2md}];
upa = Table[u[i]'[t], {i, 1, 2md}];
ics = Flatten[{Table[u[i][0] == 0, {i, 1, md}], u[md + 1][0] == v,
Table[u[i][0] == 0, {i, md + 2, 2md}]}];
eqns = Flatten[Table[upa[[i]] == (aa.ua)[[i]], {i, 1, 2md}]];
eqns = Flatten[{eqns, ics}];
MatrixForm[eqns];
soln = NDSolve[eqns, Table[u[i], {i, 1, 2md}], {t, 0, tf}];
ut = Table[Evaluate[Table[u[i][t], {i, 1, md}]/.soln], {t, 0, tf}];
Table[Evaluate[Table[u[i][t], {i, md + 1, 2md}]/.soln], {t, 0, tf}];
cc = Flatten[ut[[Dimensions[ut][[1]]]]]
```

$\{-0.537323, -2.02877, 0.514555, 0.941688, -0.584251\}$

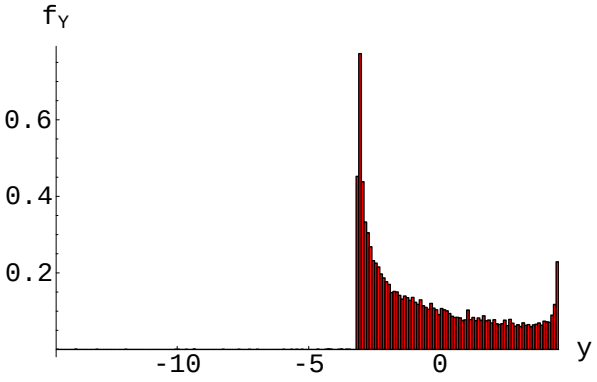
Calculate the approximated transformation.

```
as[t_] = Sum[cc[[i]]p[t, i - 1], {i, 1, md}];
Plot[as[t], {t, -6, 6}, AxesLabel → {"x", "ghat(x)"}]
```



Using approximate transformation just found, generate the pdf.

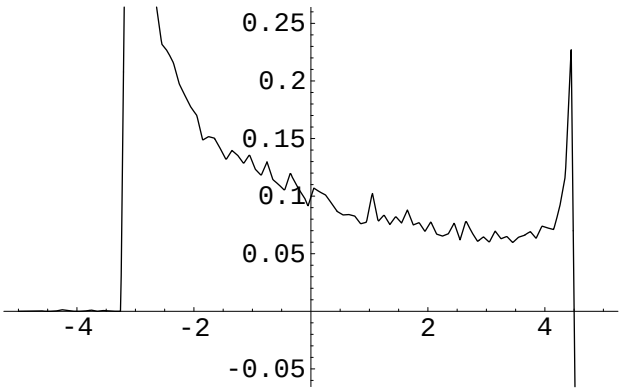
```
ss = 30000;
ri = Sort[Table[Random[NormalDistribution[0, 1]], {i, 1, ss}]];
data = Table[0, {i, 1, ss}];
For[i = 1, i ≤ ss, i++,
data[[i]] = as[ri[[i]]]
]
aa = Histogram[data, HistogramScale → 1, AxesLabel → {"y", "f_Y"}]
```



Create a pdf function by interpolating the midpoints of the top of each rectangle of the histogram.

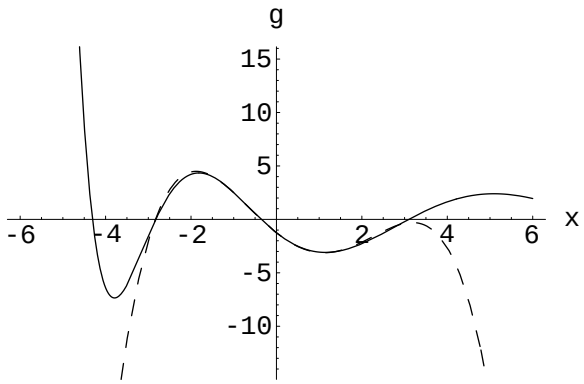
```
aaa = Transpose[Transpose[aa[[1]]][[1]]][[2]];
data1 = Table[{0, 0}, {i, 1, Length[aaa] + 2}];
data1[[1]] = {aaa[[1, 1, 1]], aaa[[1, 2, 2]]/2};
For[j = 1, j ≤ Length[aaa], j++,
```

```
ll = aaa[[j, 1]];
ur = aaa[[j, 2]];
data1[[j + 1]] = {(ll[[1]] + ur[[1]])/2, ur[[2]]}
]
data1[[Length[data1]]] = {aaa[[Length[aaa], 2]][[1]], 0};
ah = Interpolation[data1, InterpolationOrder → 1];
Plot[{ah[x]}, {x, -5, 5}]
```



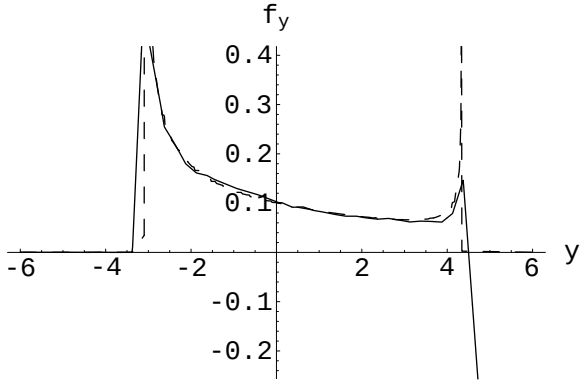
This is the actual transformation for the model problem.

```
qq[s_, t_] = 2v/Sqrt[(c/m)^2 - 4(1/m)(kb + skt)] Exp[-tc/(2m)] Sinh[t/2 Sqrt[(c/m)^2 - 4(1/m)(kb + skt)]];
q[s_] = 2v/Sqrt[(c/m)^2 - 4(1/m)(kb + skt)] Exp[-10c/(2m)] Sinh[10/2 Sqrt[(c/m)^2 - 4(1/m)(kb + skt)]];
nd = 2^8 + 1;
Plot[{q[t], as[t]}, {t, -6, 6}, AxesLabel → {"x", "g"}, PlotStyle → {GrayLevel[0], Dashing[{.03}]}]
id = Table[a + (i - 1)(b - a)/(nd - 1), {i, 1, nd}];
od = Table[0, {i, 1, nd}];
For[i = 1, i ≤ nd, i++,
od[[i]] = q[id[[i]]]
]
```



Find the pdf of the actual transformation using the transformation method as described in the previous appendix. Plot the actual and approximate pdfs and calculate the error.

```
Plot[{ah[t], pp1[t]}, {t, −6, 6}, AxesLabel → {"y", "f_y"},
PlotStyle → {GrayLevel[0], Dashing[{.03}]}]
Display["MChermite4pdf.eps", %, "EPS"]
WeightedError = Sqrt[NIntegrate[(ah[t] − pp1[t])^2, {t, −3, 4.1}]]
```



0.0323146

Bibliography

1. Wiener, N. "The Homogeneous Chaos", *American Journal of Mathematics*, 60: 897-936 (1938).

2. Silverman, B. *Density Estimation for Statistics and Data Analysis*. London: Chapman and Hall, 1986.

3. Cassella, G. and R. Berger. *Statistical Inference*. Belmont, California: Wadsworth and Brooks/Cole, 1990.

4. Xiu, Dongbin and Em Karniadakis. "Modeling Uncertainty in Flow Simulations via Generalized Polynomial Chaos", *Journal of Computational Physics*, 187: 137-167 (2003).

5. Millman, D. R. "The Fourier Chaos in Complex Exponential Form for One Uncertainty". AFIT Internal Correspondence. 2002.

6. Millman, D., P. King, P. Beran, and L. Chilton. "Estimating the Probability of Failure of a Nonlinear Aeroelastic System With Initial Condition and Parametric Uncertainties", *Journal of Aircraft*, Accepted for Publication. (February 2005).

7. Field Jr., R.V. and M. Grigoriu. "On the Accuracy of the Polynomial Chaos Approximation", *Probabilistic Engineering Mechanics*, 19: 65-80 (2004).

8. Ghanem, R. G. and P. D. Spanos. *Stochastic Finite Element Methods: A Spectral Approach*. Springer-Verlag, 1991.

9. Xiu, D., D. Lucor, H. C. Su, and E. Karniadakis. "Stochastic Modeling of Flow-Structure Interactions Using Generalized Polynomial Chaos", *Journal of Fluids Engineering*, 124: 51-58 (March 2002).

10. Reagan, M. T., N. H. Najm, J. B. Debusschere, P. O. Le Maitre, O. M. Knio, and R. G. Ghanem. "Spectral Stochastic Uncertainty Quantification in Chemical Systems". Unpublished Article, http://gmfe16.cemif.univevry.fr:8080/pub/REG04R.pdf, 20 November 2004.

11. Atkinson, Kendall and Weimin Han. *Theoretical Numerical Analysis*, Springer-Verlang, 2001.

12. Engel, David W. and Lawrence K. Chilton. "Adaptive Response Modeling for Quantifying Uncertainty in Scientific Simulations". Pacific Northwest National Laboratory Proposal Submitted to U.S. Department of Energy. August 2004.

13. Xiu, Dongbin, and Em Karniadakis. "The Wiener-Askey Polynomial Chaos for Stochastic Differential Equations", *Brown University Scientific Computing Report Series*. 2003.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From – To)*
21-03-2005	**Master's Thesis**	Jun 2003 – Mar 2004

4. TITLE AND SUBTITLE

A Sampling and Transformation Approach to Solving Random Differential Equations

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Erich, Roger, A., Second Lieutenant, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GAM/ENC/05-4

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

N/A

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This research explores an innovative sampling method used to conduct uncertainty analysis on a system with one random input. Given the distribution of the random input, X, we seek to find the distribution of the output random variable Y. When the functional form of the transformation Y=g(X) is not explicitly known, complicated procedures, such as stochastic projection or Monte Carlo simulation must be employed. The main focus of this research is determining the distribution of the random variable Y=g(X) where g(X) is the solution to an ordinary differential equation and X is a random parameter. Here, y=g(X) is approximated by constructing a sample $\{X_i, Y_i\}$ where the X_i are not random, but chosen to be evenly spaced on the interval [a, b] and $Y_i=g(X_i)$. Using this data, an efficient approximation $\hat{g}(X) \sim g(X)$ is constructed. Then the transformation method, in conjunction with $\hat{g}(X)$, is used to find the probability density function of the random variable Y. This uniform sampling method and transformation method will be compared to the stochastic projection and Monte Carlo methods currently being used in uncertainty analysis. It will be demonstrated, through several examples, that the proposed uniform sampling method and transformation method can work faster and more efficiently than the methods mentioned.

15. SUBJECT TERMS

Random Differential Equation, Uniform Sampling Method, Monte Carlo Simulation, Stochastic Projection, Polynomial Chaos

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
REPORT	**ABSTRACT**	**c. THIS PAGE**			Lawrence K. Chilton (ENC)
U	**U**	**U**	**UU**	103	**19b. TELEPHONE NUMBER** *(Include area code)* (937) 255-3636, ext 4523; e-mail: Lawrence.Chilton@afit.edu